```
import pandas as pd
import numpy as np
import math
```

## Part a

```
df=pd.read_csv("/content/hw9_part1_data.csv")
genere=np.array(df)[0,1:]
df2= df.drop(0)
print(np.array(df2))
maximum_val=np.max((np.array(df2.fillna(-10))[:,1:]).astype(np.float))
minimum_val=np.min((np.array(df2.fillna(10000))[:,1:]).astype(np.float))

all=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)
all_notnan = all[all>-100000000000]

average=np.mean(all_notnan)
print("Maximum review value",maximum_val)
print("Minimum review value",minimum_val)
print("The overall mean is",np.round(np.mean(all_notnan),2))
```

```
<ipython-input-172-36144bda3164>:1: DtypeWarning: Columns (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,
  df=pd.read_csv("/content/hw9_part1_data.csv")
[[2 '3.5' '3.5' ... nan nan nan]
 [3 '2.5' nan ... nan nan nan]
 [4 '5' '4.5' ... nan nan nan]
 ...
 [2669 3.0 3.0 ... nan nan nan]
 [2670 4.5 nan ... nan nan 1.0]
 [2671 4.5 2.5 ... nan nan nan]]
<ipython-input-172-36144bda3164>:5: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  maximum_val=np.max((np.array(df2.fillna(-10))[:,1:]).astype(np.float))
<ipython-input-172-36144bda3164>:6: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  minimum_val=np.min((np.array(df2.fillna(10000))[:,1:]).astype(np.float))
Maximum review value 5.0
Minimum review value 0.5
The overall mean is 3.3
<ipython-input-172-36144bda3164>:8: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  all=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)
```

## Answer

1. What is the highest review given? **Maximum review value is 5.0**

2. What is the lowest review?

**Minimum review value is 0.5**

3. What is the overall average review? **The overall average review is 3.3**

## Part b

```
movies=np.array(df2.columns)[1:]
p_1460=np.array(df2)[1460,1:]
g_1460=genere[p_1460==5]
print("Generes person 1462 ",g_1460)

print("Movies person 1462",movies[p_1460==5] )

p_45=(np.array(df2)[43,1:]).astype(np.float)
```

```
g_45=genere[p_45==5]
print("Generes person 45 ",g_45)
print("Movies person 45",movies[p_45==5] )
```

```
Generes person 1462  ['Crime|Drama|Thriller' 'Action|Drama|Romance|War' 'Children|Drama'
 'Drama|Romance' 'Drama|Romance' 'Drama' 'Drama|Musical'
 'Children|Drama|Fantasy' 'Adventure|Drama' 'Drama|Romance'
 'Action|Drama|Sci-Fi']
Movies person 1462 ['Taxi Driver (1976)' 'Rob Roy (1995)' 'Little Princess, A (1995)'
 "William Shakespeare's Romeo + Juliet (1996)" 'Quiet Man, The (1952)'
 'Raging Bull (1980)' 'Pink Floyd: The Wall (1982)'
 'Field of Dreams (1989)' 'Man Who Would Be King, The (1975)'
 'Playing by Heart (1998)' 'War of the Worlds, The (1953)']
Generes person 45  ['Comedy|Drama' 'Comedy|Crime' 'Comedy|Fantasy|Romance'
 'Adventure|Comedy|Sci-Fi' 'Comedy|Drama']
Movies person 45 ['Jack (1996)' 'Sting, The (1973)' 'Groundhog Day (1993)'
 'Howard the Duck (1986)' 'Anywhere But Here (1999)']
<ipython-input-173-701ae6e28e8f>:8: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  p_45=(np.array(df2)[43,1:]).astype(np.float)
```

```
print(genere[4])
```

```
Comedy
```

## Answer

What genre of movie does this individual tend to give review scores of 5? **The user tends to give review scores of 5 to drama movies.**

How does this individual differ individual 45 ? **User 1462 prefers genre drama and this can be seen from their highest reviews of movies such as movie "Taxi Driver" and movie "Field of Dreams". However, user 45 seems to prefer Genre Comedy, as seen from their 5-star reviews of movie "Howard the Duck" and movie "Jack".**

What type of movie does this individual rate highly? **The user 45 tends to like comedy movies**

# Part c

## Answer

**I would expect to separate people by their preferences. I would expect that I can separate people by the following genere preferences: "comedy, romance,thriller, drama,horror,Sci-Fi and Adventure". Hence I would expect 7 type of people according to the genere.**

# Part d

```
all_wz=(np.array(df2.fillna(0))[:,1:]).astype(np.float)

u,s,vt=np.linalg.svd(all_wz)
```

```
k=7
data_reco=u[:,0:k].dot(np.diag(s[0:k])).dot(vt[0:k,:])

jumanji_p45=data_reco[43,1]

print(np.round(jumanji_p45,2))
```

```
<ipython-input-174-d4e032d1d758>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  all_wz=(np.array(df2.fillna(0))[:,1:]).astype(np.float)
1.42
```

```
sum_squared=0
for i in range(all.shape[0]):
```

```
    for j in range(all.shape[1]):
      if all[i,j]!=-100000000000:
        sum_squared=sum_squared+(all[i,j]-data_reco[i,j])*(all[i,j]-data_reco[i,j])

s_r_m_s_d=np.sqrt(sum_squared/(all.shape[0]*all.shape[1]))

print("Mean squared distance",np.round(s_r_m_s_d,2))
```

```
Mean squared distance 0.81
```

## Answer

1. Report the estimate of user 45 (index 43) review of the movie Jumanji **Estimate for user 45 for Jumanji is 1.42**
2. Compute and report the average difference between the true (non-missing) values in the review matrix and the reconstructed matrix. **The square root of the Mean squared distance is 0.81**

# Part e

```
count_c=[]
means=[]
print(all)
for j in range(all.shape[1]):
  count_c.append(0)
  for i in range(all.shape[0]):
    if all[i,j]==-100000000000:
      all[i,j]=0
      count_c[j]=count_c[j]+1
  means.append(np.sum(all[:,j])/(all.shape[0]-count_c[j]))
print(all)
```

```
[[ 3.5e+00  3.5e+00  3.0e+00 ... -1.0e+11 -1.0e+11 -1.0e+11]
 [ 2.5e+00 -1.0e+11 -1.0e+11 ... -1.0e+11 -1.0e+11 -1.0e+11]
 [ 5.0e+00  4.5e+00 -1.0e+11 ... -1.0e+11 -1.0e+11 -1.0e+11]
 ...
 [ 3.0e+00  3.0e+00  3.5e+00 ... -1.0e+11 -1.0e+11 -1.0e+11]
 [ 4.5e+00 -1.0e+11 -1.0e+11 ... -1.0e+11 -1.0e+11  1.0e+00]
 [ 4.5e+00  2.5e+00  5.0e-01 ... -1.0e+11 -1.0e+11 -1.0e+11]]
[[3.5 3.5 3.  ... 0.  0.  0. ]
 [2.5 0.  0.  ... 0.  0.  0. ]
 [5.  4.5 0.  ... 0.  0.  0. ]
 ...
 [3.  3.  3.5 ... 0.  0.  0. ]
 [4.5 0.  0.  ... 0.  0.  1. ]
 [4.5 2.5 0.5 ... 0.  0.  0. ]]
```

```
print(means)
```

```
[3.8681705298013247, 3.0228690228690227, 2.773361976369495, 2.4674657534246576, 2.5741525423728815, 3.790623335109217, 2.910
```

```
all_m=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)

for j in range(all_m.shape[1]):
  for i in range(all_m.shape[0]):
    if all_m[i,j]==-100000000000:
      all_m[i,j]=means[j]
```

```
<ipython-input-178-e6e72a5d8d2c>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  all_m=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)
```

```
u,s,vt=np.linalg.svd(all_m)


k=7
data_reco_m=u[:,0:k].dot(np.diag(s[0:k])).dot(vt[0:k,:])

jumanji_p45=data_reco_m[43,1]

print(np.round(jumanji_p45,2))
```

```
    2.47
```

```
all_m2=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)

sum_squared_m=0
for i in range(all_m2.shape[0]):
  for j in range(all_m2.shape[1]):
    if all_m2[i,j]!=-100000000000:
      sum_squared_m=sum_squared_m+(all_m2[i,j]-data_reco_m[i,j])*(all_m2[i,j]-data_reco_m[i,j])

s_r_m_s_dM=np.sqrt(sum_squared_m/(all_m2.shape[0]*all_m2.shape[1]))

print("Mean squared distance",np.round(s_r_m_s_dM,2))
```

```
    <ipython-input-180-eb85f982431f>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
    Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
      all_m2=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)
    Mean squared distance 0.35
```

## Answer

1. Report the estimate of user 45 (index 43) review of the movie Jumanji. **Estimate for user 45 for Jumanji is 2.47**
2. Compute and report the average difference between the true (non-missing) values in the review matrix and the reconstructed matrix.
   **Square root of the mean squared distance is 0.35**

# Part 2 of homework

## Part f

```
count_cf=[]
meansf=[]
allf=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)

for i in range(allf.shape[0]):
  count_cf.append(0)
  for j in range(allf.shape[1]):
    if allf[i,j]==-100000000000:
      allf[i,j]=0
      count_cf[i]=count_cf[i]+1
  meansf.append(np.sum(allf[i,:])/(allf.shape[0]-count_cf[i]))
print(allf)
```

```
    <ipython-input-192-a37295e6f016>:3: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
    Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
      allf=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)
    [[3.5 3.5 3.  ... 0.  0.  0. ]
     [2.5 0.  0.  ... 0.  0.  0. ]
     [5.  4.5 0.  ... 0.  0.  0. ]
     ...
     [3.  3.  3.5 ... 0.  0.  0. ]
     [4.5 0.  0.  ... 0.  0.  1. ]
     [4.5 2.5 0.5 ... 0.  0.  0. ]]
```

```
print(meansf)
```

```
[1.1376712328767122, 1.042675159235669, 1.241958041958042, 1.5271565495207668, 1.8567046450482034, 1.5486358244365361, 1.742
```

```
all_mf=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)

for i in range(all_mf.shape[0]):
  for j in range(all_mf.shape[1]):
    if all_mf[i,j]==-100000000000:
      all_mf[i,j]=meansf[i]
```

```
<ipython-input-194-1ccc732c8c15>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  all_mf=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)
```

```
u,s,vt=np.linalg.svd(all_mf)


k=7
data_reco_mf=u[:,0:k].dot(np.diag(s[0:k])).dot(vt[0:k,:])

jumanji_p45=data_reco_mf[43,1]

print(np.round(jumanji_p45,2))
```

```
1.85
```

```
all_m2f=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)

sum_squared_mf=0
for i in range(all_m2f.shape[0]):
  for j in range(all_m2f.shape[1]):
    if all_m2f[i,j]!=-100000000000:
      sum_squared_mf=sum_squared_mf+(all_m2f[i,j]-data_reco_mf[i,j])*(all_m2f[i,j]-data_reco_mf[i,j])

s_r_m_s_dMf=np.sqrt(sum_squared_mf/(all_m2f.shape[0]*all_m2f.shape[1]))

print("Mean squared distance",np.round(s_r_m_s_dMf,2))
```

```
<ipython-input-196-bfcf1ba7e2b0>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  all_m2f=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)
Mean squared distance 0.52
```

## Answer

1. Report the estimate of user 45 (index 43) review of the movie Jumanji. **Estimate for user 45 for Jumanji is 1.85**
2. Additionally, compute and report the average difference between the true (non-missing) values in the review matrix and the reconstructed matrix. **The Square root of the mean squared distance 0.52**

## Part g

## Answer

1. Interpret the meaning of each method of replacing missing values used in the previous subquestions.
   By replacing missing values with row average the interpretation is: **If we believe that a ranking of a movie is primarily a function of the user, then we might expect missing values to be similar to other values given by the user.**

   By replacing missing values with column average the interpretation is: **If we believe that a ranking of a movie is primarily a function of the movie, then we might expect the missing values to be similar to other rankings given to the movie**

   By replacing missing values with zeros is: **If users only rank movies they have watched, then we might expect the missing values to be low compared to their counterparts due to self selection of movies**
2. Which method is the most effective at reducing the difference between true and reconstructed values in the matrix? **We can compare the square root of mean squared distance obtained in each case. Filling with zeros: Square root of the mean squared distance 0.81, Column average: Square root of the mean squared distance 0.35, Row average: 0.52.**

**Hence, the column average is the most effective at reducing the difference between true and reconstructed values in the matrix**

3. Give another method that could've been used, and explain what effect it would have. **Replace missing ranking values of movies of certain user "A" by the mean of users that have originally watched the movie that user "A" has not watched and that have similar movie preferences of user "A". For example, we could firstly see the preferences of user "A". Suppose is comedy to show how this would work. Then we collect all other users that have watched the movie that A has not watched. Then we filter only those other users that have liked many comedy movies ( for example could be 70% of movies to be rank 5 to be comedy). Finally we take the mean of the rankings of the filtered users and replace the missing value by that mean. We itereate over all missing values implementing these steps. This algorithm would have a better prediction, because we are considering the preference of the people that are similar to "A" in movie preferences, so there is higher probability to fill values correctly as "A" would have ranked.**

## Part h

```
allh=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)

count_ch=[]
meansh=[]
print(allh)
for j in range(allh.shape[1]):
  count_ch.append(0)
  for i in range(allh.shape[0]):
    if allh[i,j]==-100000000000:
      allh[i,j]=0
      count_ch[j]=count_ch[j]+1
  meansh.append(np.sum(allh[:,j])/(allh.shape[0]-count_ch[j]))
```

```
<ipython-input-197-519ea04fa96d>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  allh=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)
[[ 3.5e+00  3.5e+00  3.0e+00 ... -1.0e+11 -1.0e+11 -1.0e+11]
 [ 2.5e+00 -1.0e+11 -1.0e+11 ... -1.0e+11 -1.0e+11 -1.0e+11]
 [ 5.0e+00  4.5e+00 -1.0e+11 ... -1.0e+11 -1.0e+11 -1.0e+11]
 ...
 [ 3.0e+00  3.0e+00  3.5e+00 ... -1.0e+11 -1.0e+11 -1.0e+11]
 [ 4.5e+00 -1.0e+11 -1.0e+11 ... -1.0e+11 -1.0e+11  1.0e+00]
 [ 4.5e+00  2.5e+00  5.0e-01 ... -1.0e+11 -1.0e+11 -1.0e+11]]
```

```
print(meansh)
```

```
[3.8681705298013247, 3.0228690228690227, 2.773361976369495, 2.4674657534246576, 2.5741525423728815, 3.790623335109217, 2.910
```

```
all_mh=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)

for j in range(all_mh.shape[1]):
  for i in range(all_mh.shape[0]):
    if all_mh[i,j]==-100000000000:
      all_mh[i,j]=meansh[j]
```

```
<ipython-input-199-1f5608fa3dda>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  all_mh=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)
```

```
from sklearn.decomposition import NMF
k=7

model = NMF(n_components=k, init='random', random_state=0,max_iter = 200)
W = model.fit_transform(all_mh)
H=model.components_


data_reco_h=W.dot(H)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/decomposition/_nmf.py:1665: ConvergenceWarning: Maximum number of iterations 2
  warnings.warn(
```

```
print(data_reco_h.shape)
```

```
(2670, 2153)
```

```
print(data_reco_h)
```

```
[[3.96274553 3.5493988  3.06847111 ... 3.30142748 2.89357668 3.69019967]
 [3.40273205 2.72172648 2.60930045 ... 3.33654162 2.94382935 3.5859238 ]
 [4.4890659  3.37545775 2.92347241 ... 3.30914281 2.85130937 3.75915754]
 ...
 [3.13116946 2.88505109 2.65236541 ... 3.41444298 2.89237944 3.69052953]
 [3.81832982 3.36125351 3.02122253 ... 3.2812619  2.8620062  3.45233566]
 [3.50981848 3.4258518  3.05722695 ... 3.20729636 2.79700068 3.3931038 ]]
```

```
print(all_mh.shape)
```

```
(2670, 2153)
```

```
jumanji_p45=data_reco_h[43,1]

print(np.round(jumanji_p45,2))
```

```
2.44
```

```
all_m2h=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)

sum_squared_mh=0
for i in range(all_m2h.shape[0]):
  for j in range(all_m2h.shape[1]):
    if all_m2h[i,j]!=-100000000000:
      sum_squared_mh=sum_squared_mh+(all_m2h[i,j]-data_reco_h[i,j])*(all_m2h[i,j]-data_reco_h[i,j])

s_r_m_s_dMh=np.sqrt(sum_squared_mh/(all_m2h.shape[0]*all_m2h.shape[1]))

print("Mean squared distance",np.round(s_r_m_s_dMh,2))
```

```
<ipython-input-205-1ad3ab7213a2>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  all_m2h=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)
Mean squared distance 0.36
```

## Answer

1. Report the estimate of user 45 (index 43) review of the movie Jumanji. **Estimate for user 45 for Jumanji is 2.44.**
2. Compute and report the average difference between the true (non-missing) values in the review matrix and the reconstructed matrix **The square root of mean squared distance 0.36**
3. How does this compare to the average difference found using SVD? **When using SVD, I obtained a square root of mean squared distance of 0.35, while using NMF I obtained 0.36 . So, using NMF there is slightly more error in reconstructed true(non-missing) values.**

# Part i

## Answer

1. Explain a reason why one would choose to use SVD over NMF for this data. **The square root of mean squared distance for SVD is smaller than NMF. There is less error in the estimates for the original non missing values.**
2. Explain a reason why one would choose to use NMF over SVD for this data. **NMF basis vectors could have more interpretability.**
3. Which would you recommend? **I would recommend NMF, because even though the square root of mean squared distance is slightly greater, the interpretability of NMF could give hints about the users and their movie preferences. With the interpretability of the basis vectors of NMF, further analysis could be done by applying algorithms that implement clustering afterwards.**

# Part j

```
allj=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)

count_cj=[]
meansj=[]
for j in range(allj.shape[1]):
  count_cj.append(0)
  for i in range(allj.shape[0]):
    if allj[i,j]==-100000000000:
      allj[i,j]=0
      count_cj[j]=count_cj[j]+1
  meansj.append(np.sum(allj[:,j])/(allj.shape[0]-count_cj[j]))
```

```
<ipython-input-206-1345ff058a35>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  allj=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)
```

```
print(meansh)
```

```
[3.8681705298013247, 3.0228690228690227, 2.773361976369495, 2.4674657534246576, 2.5741525423728815, 3.790623335109217, 2.910
```

```
all_mj=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)

for j in range(all_mj.shape[1]):
  for i in range(all_mj.shape[0]):
    if all_mj[i,j]==-100000000000:
      all_mj[i,j]=meansj[j]
```

```
<ipython-input-208-ba263c63d0d4>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  all_mj=(np.array(df2.fillna(-100000000000))[:,1:]).astype(np.float)
```

```
from sklearn.decomposition import NMF
k=5

modelj = NMF(n_components=k, init='random', random_state=0,max_iter = 200)
Wj = modelj.fit_transform(all_mj)
Hj=modelj.components_


data_reco_j=Wj.dot(Hj)

print(data_reco_j)
```

```
[[4.00155487 3.53288518 3.0910703  ... 3.34833623 2.91702183 3.65968624]
 [3.29396909 2.74142841 2.63692517 ... 3.37084738 2.94876025 3.57436134]
 [4.48486194 3.32359991 2.93176706 ... 3.33316791 2.87836959 3.74652253]
 ...
 [3.52850192 2.96025829 2.65585915 ... 3.37829422 2.90294417 3.64650869]
 [3.66024902 3.38854417 2.8756275  ... 3.30140809 2.82255107 3.58017199]
 [3.74678506 3.46317634 2.89864755 ... 3.15871554 2.72558821 3.48596283]]
/usr/local/lib/python3.9/dist-packages/sklearn/decomposition/_nmf.py:1665: ConvergenceWarning: Maximum number of iterations 2
  warnings.warn(
```

```
print(type(data_reco_j))
```

```
<class 'numpy.ndarray'>
```

```
movies=np.array(df2.columns)[1:]
p_1460j=data_reco_j[1460]

p_1460F=(p_1460).astype(np.float)


recommendations=[]
rankings=[]
generes=[]
for i in range(p_1460j.shape[0]):
  if (p_1460j[i]>=3.92 and np.isnan(p_1460F[i])):
    recommendations.append(movies[i])
    rankings.append(p_1460j[i])
    generes.append(genere[i])
```

```
print("Recommendations",recommendations, "with rankings",rankings,"with generes",generes)
```

```
mendations ['Nine to Five (a.k.a. 9 to 5) (1980)', 'K-PAX (2001)', 'Before Sunset (2004)'] with rankings [3.936377964658053,
hon-input-211-5bf6199e3bfd>:4: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this
cated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
460F=(p_1460).astype(np.float)
```

```
p_1460F4_generes4=genere[p_1460F==4]

print(p_1460F4_generes4)

counter_comedy_crime=0
for i in range(p_1460F4_generes4.shape[0]):
  if (p_1460F4_generes4[i].__contains__('Comedy') or p_1460F4_generes4[i].__contains__('Crime')):
    counter_comedy_crime=counter_comedy_crime+1

print("Percentage of ranked 3, 4 commedy",counter_comedy_crime/p_1460F4_generes4.shape[0]*100)
```

```
['Adventure|Animation|Children|Comedy|Fantasy' 'Comedy|Crime|Thriller'
 'Crime|Mystery|Thriller' 'Comedy' 'Comedy|Crime|Drama|Thriller'
 'Action|Crime|Fantasy|Thriller' 'Children|Drama'
 'Comedy|Romance|Thriller' 'Animation|Children|Drama|Fantasy|Musical'
 'Animation|Children|Fantasy|Musical' 'Adventure|Drama|Sci-Fi'
 'Comedy|Drama' 'Drama' 'Horror' 'Drama|Mystery|Sci-Fi' 'Drama|War'
 'Crime|Drama|Film-Noir|Thriller' 'Crime|Film-Noir|Thriller'
 'Drama|Western' 'Action|Adventure|Animation|Sci-Fi' 'Comedy'
 'Drama|Romance' 'Drama' 'Drama|Romance' 'Crime|Drama|Romance' 'Comedy'
 'Horror|Thriller' 'Horror|Thriller'
 'Action|Adventure|Children|Comedy|Fantasy' 'Action|Adventure|Sci-Fi'
 'Comedy|Fantasy|Romance' 'Action|Comedy|Crime|Thriller' 'Drama'
 'Horror|Thriller' 'Children|Comedy' 'Crime|Thriller'
 'Action|Adventure|Thriller' 'Comedy|Drama|Musical' 'Drama|War'
 'Adventure|Drama' 'Adventure|Comedy' 'Drama|Romance' 'Action|Comedy'
 'Drama|Romance|War' 'Comedy|Romance' 'Action|Crime|Thriller'
 'Documentary' 'Comedy|Romance']
Percentage of ranked 3, 4 commedy 50.0
```

## Answer

**The recommended movies are Nine to Five,K-PAX and Before Sunset.**

How did you choose these recommendations? **I picked the three movies with highest estimated ranking.** Do these recommendations align with what you'd expect them to enjoy? **Two out of the three recommendations align to what is expected from user 1462 to enjoy the most. One of the movies is expected to enjoy (ranking 4) but not the most.**

Why or why not?

**The movies K-PAX and Before Sunset are drama movies, while Nine to Five is commedy movie . The user 1462 tends to like the most drama movies, because all original ranked movies of user 1462 with rank 5 were drama movies. So, the two movies with highest estimated rank correctly predicted what would the user tend to like the most. The third recommendation, with the lowest estimated ranking among the top is a commedy|crime movie and the user 1462 originally ranked as 4 the 50% of movies that where either comedy or crime. So, as the last suggestion with lowest estimated ranking among the top, there is a 50% chance the user will like it.**

# Part k

```
all_k_wNans=(np.array(df2.fillna(-1000))[:,1:]).astype(np.float)
```

```
<ipython-input-146-64fcec8ba2f3>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  all_k_wNans=(np.array(df2.fillna(-1000))[:,1:]).astype(np.float)
```

```
all_k_wZeros=(np.array(df2.fillna(0))[:,1:]).astype(np.float)
```

```
Z=all_k_wZeros

Y=np.empty([all_k_wZeros.shape[0], all_k_wZeros.shape[1]], dtype=float)


k=5
lamda=100

for i in range(100):
  for r in range(all_k_wNans.shape[0]):
      for s in range(all_k_wNans.shape[1]):
        if all_k_wNans[r,s]==-1000:
          Y[r,s]=Z[r,s]
        if all_k_wNans[r,s]!=-1000:
          Y[r,s]=all_k_wNans[r,s]

  u,s,vt=np.linalg.svd(Y)
  s_top5=s[0:k]
  sigma_lambda=[]
  for  j in range(s_top5.shape[0]):
    sigma_lambda.append(s_top5[j]-lamda)
    if sigma_lambda[j]<=0:
      sigma_lambda[j]=0
  Z=u[:,0:k].dot(np.diag(sigma_lambda)).dot(vt[0:k,:])
```

```
<ipython-input-156-c09b9eb67c07>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  all_k_wZeros=(np.array(df2.fillna(0))[:,1:]).astype(np.float)
```

```
print(Y[43])

print(all_k_wNans[43])
```

```
[2.5        2.21066152 2.00447096 ... 1.20038246 1.05363933 2.92512831]
[    2.5 -1000.  -1000.  ... -1000.  -1000.  -1000. ]
```

```
movies=np.array(df2.columns)[1:]
p_45k=Y[43]

p_45Fk=(p_45).astype(np.float)

recommendationsk=[]
rankingsk=[]
generesk=[]
for i in range(p_45k.shape[0]):
  if (p_45k[i]>=3.471 and np.isnan(p_45Fk[i])):
    recommendationsk.append(movies[i])
    rankingsk.append(p_45k[i])
    generesk.append(genere[i])

print("Recommendations",recommendationsk, "with rankings",rankingsk,"with generes",generesk)

print("3 ranked original",genere[p_45Fk==3])
print("-----")
print("4 ranked original",genere[p_45Fk==4])
```

```
s ['2001: A Space Odyssey (1968)', 'Ghost and Mrs. Muir, The (1947)', 'M (1931)'] with rankings [3.550014425817391, 3.5849811
nal ['Drama|Romance' 'Drama|Romance' 'Drama' 'Drama|Romance'
|Romance' 'Crime|Drama|Romance' 'Comedy|Drama'
ce' 'Drama' 'Drama|Romance' 'Comedy'
|Sci-Fi|Thriller' 'Action|Thriller|Western'
|Drama|Thriller' 'Comedy|Drama|Romance' 'Comedy|Sci-Fi'
' 'Comedy|Drama|Romance' 'Adventure|Western'
Romance|Thriller' 'Adventure|Children|Fantasy|Musical'
y' 'Film-Noir|Romance|Thriller' 'Adventure|Drama'
' 'Documentary' 'Action|Adventure|Western' 'Drama'
y|Sci-Fi' 'Action|Drama|War' 'Drama|Fantasy'
ture|Drama|War' 'Drama' 'Animation|Children|Fantasy|Musical'
ce' 'Horror|Thriller' 'Action|Adventure|Sci-Fi|Thriller'
rn' 'Comedy' 'Action|Comedy' 'Comedy|Drama' 'Drama|Romance'
edy|Fantasy' 'Drama|Romance' 'Comedy' 'Horror|Thriller'
ildren|Fantasy' 'Crime|Drama' 'Crime|Drama|Thriller'
|Drama' 'Adventure|Children|Drama' 'Action|Drama|War'
e' 'Comedy|Crime' 'Drama|Romance|Sci-Fi|Thriller'
ce' 'Comedy' 'Mystery|Thriller' 'Drama' 'Comedy|Horror'
dy' 'Comedy|Romance' 'Action|Adventure|Comedy|Thriller'
edy|Drama|Musical' 'Action|Horror|Thriller'
```

```
|Romance' 'Adventure|Children|Comedy' 'Drama']

nal ['Action|Crime|Thriller' 'Crime|Drama' 'Mystery|Sci-Fi|Thriller'
ture|Comedy|Crime' 'Action|Adventure|Mystery|Sci-Fi'
er' 'Crime|Drama' 'Comedy|Crime|Drama|Thriller'
ture|Children|Comedy|Fantasy|Sci-Fi' 'Documentary' 'Drama'
ture|Sci-Fi|Thriller' 'Comedy|Crime' 'Children|Comedy'
|Romance' 'Adventure|Drama|War'
Film-Noir|Thriller' 'Comedy|Drama|Romance'
r|Sci-Fi' 'Comedy|Drama' 'Crime|Film-Noir|Mystery'
e' 'Drama|Romance' 'Action|Horror' 'Drama|Romance'
ture|Comedy|Thriller' 'Drama|Romance' 'Drama|Romance'
imation|Children|Drama|Musical' 'Action|Crime|Thriller'
|Thriller' 'Crime|Drama|Mystery|Thriller' 'Horror' 'Horror'
imation|Children|Musical' 'Drama|Sci-Fi'
|Fantasy' 'Action|Sci-Fi|War' 'Comedy' 'Comedy|Crime'
ture|Drama|Thriller' 'Action|Drama'
ture|Children|Fantasy|Mystery|Thriller' 'Comedy'
' 'Comedy|Romance' 'Action|Mystery' 'Comedy'
Mystery|Romance|Thriller' 'Action|Adventure|Thriller'
tery|Romance|Thriller' 'Drama|Mystery' 'Drama' 'Drama'
a' 'Action|Comedy' 'Drama|Romance' 'Crime|Film-Noir'
medy' 'Action|Adventure|Romance|Thriller'
ture|Drama|War' 'Action|Drama|Romance'
Thriller' 'Comedy|Romance' 'Action|Horror|Sci-Fi']
-171-f16fa7ca5bcd>:4: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning,
NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
).astype(np.float)
```

## Answer

1. Using this matrix, give three recommendations of movies for user 45 that they have not seen.

**Recommended movies: '2001: A Space Odyssey (1968)', 'Ghost and Mrs. Muir, The (1947)', 'M (1931)'**

2. How did you choose these recommendations?

**"I picked the 3 movies with the highest estimated ranking"**

3. Do these recommendations align with what you'd expect them to enjoy? Why or why not?

**The estimation ranking value is correct according to what the user 45 chose before. The drama movies originally ranked by user 45 were give a rank of 3 and the matrix completion etimated almost 3 to those movies. The recommended movies WILL NOT BE the favorite ones of user 45 , however are the top among all not seen movies. The movies will not be of their highest preference because they are estimated a ranking between 3 and 4.**