In [1]:
```
import pandas as pd
import numpy as np
from tqdm import tqdm
from sklearn.linear_model import LinearRegression
from matplotlib import pyplot as plt
```

In [2]:
```
df = pd.read_csv('movieReplicationSet(1).csv')
mean_of_user = df.iloc[:, :400].mean(axis=1, skipna=True)
mean_of_movie = df.iloc[:, :400].mean(axis=0, skipna=True)
for i in range(df.shape[0]):
    for j in range(400):
        if np.isnan(df.iloc[i, j]):
            df.iloc[i, j] = (mean_of_user[i] + mean_of_movie[j]) / 2

# user 896 has no ratings
df = df.drop(896, axis=0)
```

In [3]:
```
cod = np.zeros((400, 400))
for movie_idx in tqdm(range(400)):
    for other_movie in range(400):
        if other_movie == movie_idx:
            continue
        train_X = df.iloc[:, other_movie]  # 1096
        train_Y = df.iloc[:, movie_idx]  # 1096
        reg = LinearRegression().fit(train_X.values.reshape(-1, 1), tr
        cod[movie_idx, other_movie] = reg.score(train_X.values.reshape
```

```
100%|██████████| 400/400 [01:49<00:00,  3.65it/s]
```

In [5]:
```
movies = df.columns[:400]
```

In [7]:
```
selected_movies_idxs = cod.max(axis=1).argsort()[185:215]
selected_movies = movies[selected_movies_idxs]
candidate_idxs = np.array(list(set(range(400)) - set(selected_movies_i
feature_movies_idxs = candidate_idxs[np.random.choice(candidate_idxs.s
feature_movies = movies[feature_movies_idxs]

X = df.iloc[:, feature_movies_idxs].values
Y = df.iloc[:, selected_movies_idxs].values
# 80/20 random split
train_idxs = np.random.choice(df.shape[0], int(df.shape[0] * 0.8), rep
test_idxs = np.array(list(set(range(df.shape[0])) - set(train_idxs)))
train_X = X[train_idxs]
train_Y = Y[train_idxs]
test_X = X[test_idxs]
test_Y = Y[test_idxs]
```

In [17]:
```
#3

from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import mean_squared_error

pd.set_option('display.max_colwidth', None)

alpha_to_rmse = dict()
alpha_to_beta = dict()
for alpha in tqdm(np.arange(1, 100, 1)):
    tmp_df = pd.DataFrame()
    for i in range(30):
        reg = Ridge(alpha=alpha).fit(train_X, train_Y[:, i])  # betas
        tmp_df.loc[i, 'target movie'] = selected_movies[i]
        tmp_df.loc[i, 'RMSE'] = mean_squared_error(test_Y[:, i], reg.p
        tmp_df.loc[i, 'COD'] = reg.score(test_X, test_Y[:, i])
        tmp_df.loc[i, 'beta'] = str(reg.coef_)
    # print('alpha =', alpha, 'average RMSE =', tmp_df['RMSE'].mean())
    alpha_to_rmse[alpha] = tmp_df['RMSE'].mean()
    alpha_to_beta[alpha] = tmp_df['beta'].values
    # display(tmp_df)
print('best alpha', min(alpha_to_rmse, key=alpha_to_rmse.get))
print('best beta', alpha_to_beta[min(alpha_to_rmse, key=alpha_to_rmse.
```

```
100%|██████████| 99/99 [00:07<00:00, 14.01it/s]
```

```
best alpha 36
best beta ['[0.27540373 0.08764685 0.04177319 0.09743275 0.07991411
0.07459153\n 0.00581997 0.05936197 0.01237043 0.05756957]'
 '[ 0.23193362  0.04645869  0.05737426  0.18511422  0.11273515  0.02
795966\n -0.00131961  0.04563403  0.06168206  0.05942398]'
 '[0.11320316 0.05866095 0.01808954 0.22749806 0.21947683 0.07706078
\n 0.03226618 0.06575423 0.10091374 0.06476627]'
 '[0.17397535 0.0412683  0.02821257 0.23037068 0.26265426 0.04807646
\n 0.0365501  0.05242761 0.11355068 0.10117706]'
 '[ 0.13821916  0.09616787 -0.04028961  0.23395584  0.1919155   0.03
616793\n  0.02074936  0.00027284  0.11928622  0.04272582]'
 '[0.16516807 0.12936614 0.01002407 0.14815316 0.05559277 0.04527951
\n 0.0597851  0.02674717 0.05827291 0.04495979]'
 '[0.11409997 0.07835787 0.02332109 0.1835247  0.1714847  0.02394124
\n 0.03876157 0.06917344 0.0285601  0.07532343]'
 '[0.29395589 0.05160622 0.00653949 0.01406416 0.23843663 0.04407387
\n 0.01295565 0.0502702  0.05576813 0.06506818]'
 '[0.19464127 0.09072303 0.01516825 0.15030911 0.11486133 0.02069522
\n 0.0395127  0.01474578 0.09679885 0.05256486]'
 '[0.13885097 0.1055441  0.06574639 0.20399883 0.1049835  0.02112654
\n 0.04169164 0.03086192 0.01607419 0.046982  ]'
 '[ 0.10796176  0.03755366  0.09044681  0.06485882  0.29830267  0.01
086905\n -0.00378714  0.0487325   0.03276507  0.12844074]'
 '[ 0.10369676  0.10629452  0.01440439 -0.01095554  0.27479238  0.11
103829\n  0.02231255  0.00249096  0.05733368  0.07998716]'
 '[ 0.22356695  0.09302971  0.0322176   0.17703413  0.12451654  0.06
008587\n  0.02336353  0.03396192  0.03886774 -0.00898165]'
 '[0.12460599 0.08128411 0.06330377 0.07237005 0.21521904 0.05659689
```

```
\n 0.04574707 0.04763059 0.0325287  0.06933648]'
 '[ 0.22182195  0.08887876  0.03468051  0.19746261  0.23608273 -0.03
391944\n  0.05798037  0.00944559  0.06858123  0.04634378]'
 '[0.07861118 0.08111483 0.12596278 0.14625924 0.24720374 0.00455786
\n 0.11934981 0.09432717 0.06856199 0.05128079]'
 '[-0.02512949  0.08111158  0.17367689  0.03615305  0.25766113  0.03
886156\n  0.06334568  0.12636933  0.06620493  0.11555253]'
 '[0.12615266 0.0896007  0.02169502 0.15901395 0.17327258 0.02907719
\n 0.03231623 0.03593598 0.05124057 0.01156284]'
 '[0.30430526 0.05618969 0.02439172 0.04599458 0.07971465 0.06461299
\n 0.01637845 0.08065363 0.01671786 0.08192025]'
 '[0.0600069  0.04830947 0.03486677 0.18385962 0.19257334 0.06079824
\n 0.02845586 0.02075501 0.02434861 0.09144478]'
 '[0.25612172 0.04160295 0.02423072 0.11347432 0.10666645 0.0076274
\n 0.02806004 0.07993726 0.04711846 0.01430153]'
 '[ 0.197982    0.13479999 -0.02658617  0.13746845  0.21567261  0.04
753241\n  0.02006443  0.0143684   0.04495706  0.0635622 ]'
 '[0.21872684 0.08088145 0.02461118 0.13878466 0.18566911 0.0276311
\n 0.02000326 0.01191844 0.03982044 0.02346613]'
 '[ 0.25140387  0.02673197  0.05294079  0.01025616  0.1044961   0.05
308579\n  0.0266806   0.10917876 -0.00251249  0.02989873]'
 '[0.23564803 0.12770464 0.03162274 0.05016529 0.23200145 0.00211983
\n 0.00405794 0.01377039 0.11352297 0.08846904]'
 '[0.05766665 0.12888028 0.13989779 0.11484549 0.24272716 0.035894\n
0.0335424  0.07016222 0.03426831 0.06472139]'
 '[0.06640745 0.11912194 0.14374705 0.05320636 0.27624112 0.05277367
\n 0.05098512 0.07276778 0.09486472 0.00077893]'
 '[ 0.16606247  0.17132847 -0.01503642  0.17961863  0.13582237  0.03
```

```
284111\n  0.01006616  0.01469663  0.07460837  0.00768566]'
 '[ 0.31714559  0.06091008  0.00992368  0.13398959  0.14715227  0.00
377599\n -0.00395105  0.06874858  0.05192719  0.05672794]'
 '[ 0.19232058  0.07526525  0.00337571  0.1032591   0.28096084  0.00
657233\n  0.01984696 -0.00045116  0.08856002  0.0981605 ]']
```

In [18]:
```
#4

from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import mean_squared_error

pd.set_option('display.max_colwidth', None)

alpha_to_rmse = dict()
for alpha in tqdm(np.arange(0.001, 0.01, 0.0001)):
    tmp_df = pd.DataFrame()
    for i in range(30):
        reg = Lasso(alpha=alpha).fit(train_X, train_Y[:, i])  # betas
        tmp_df.loc[i, 'target movie'] = selected_movies[i]
        tmp_df.loc[i, 'RMSE'] = mean_squared_error(test_Y[:, i], reg.p
        tmp_df.loc[i, 'COD'] = reg.score(test_X, test_Y[:, i])
        tmp_df.loc[i, 'beta'] = str(reg.coef_)
    # print('alpha =', alpha, 'average RMSE =', tmp_df['RMSE'].mean())
    alpha_to_rmse[alpha] = tmp_df['RMSE'].mean()
    alpha_to_beta[alpha] = tmp_df['beta'].values
    # display(tmp_df)
print('best alpha', min(alpha_to_rmse, key=alpha_to_rmse.get))
print('best beta', alpha_to_beta[min(alpha_to_rmse, key=alpha_to_rmse.
```

```
100%|██████████| 90/90 [00:05<00:00, 17.82it/s]
```

```
best alpha 0.003400000000000001
best beta ['[0.35399973 0.08306072 0.03746215 0.09079057 0.07459066
0.0716424\n 0.         0.05331556 0.00243288 0.05004826]'
 '[ 0.28424257  0.01940489  0.05253768  0.22631892  0.11638925  0.01
708411\n -0.          0.03862771  0.05473605  0.05164779]'
 '[0.10728802 0.03377036 0.00568001 0.28831871 0.28549594 0.06660188
\n 0.02150733 0.05933606 0.09425776 0.05334035]'
 '[0.19030786 0.00624084 0.01469234 0.28360791 0.34927666 0.03226227
\n 0.02482976 0.04194357 0.10609162 0.09168546]'
 '[ 0.13988793  0.07936229 -0.04138152  0.29479112  0.2345007   0.02
208647\n  0.00793127 -0.          0.11500249  0.02543138]'
 '[0.19527794 0.13553795 0.00267488 0.17705919 0.03479084 0.03971591
\n 0.0547559  0.02106518 0.05533857 0.03871745]'
 '[0.11520371 0.06537877 0.01347788 0.22805724 0.21843231 0.01209633
\n 0.03092495 0.06495369 0.01582734 0.06882094]'
 '[ 0.37145699  0.02898865  0.         -0.          0.32307551  0.03
143005\n  0.00144355  0.03751338  0.04360592  0.04845934]'
 '[0.23517426 0.08307483 0.00641632 0.17447122 0.12758222 0.00976987
\n 0.0314868  0.0054951  0.09677838 0.04385027]'
 '[0.15186134 0.1000623  0.06312305 0.26240319 0.11051922 0.01082034
\n 0.03393842 0.02399719 0.00402959 0.03795552]'
 '[ 0.11017916  0.01688681  0.08558525  0.03183052  0.42869848  0.\n
-0.          0.03904622  0.01483111  0.12349844]'
 '[ 9.40255337e-02  9.95855095e-02  2.58832297e-04 -0.00000000e+00\n
3.79828170e-01  1.04639912e-01  1.10396272e-02 -0.00000000e+00\n  4.
22342870e-02  6.69747464e-02]'
 '[ 0.26965546  0.07988065  0.02287452  0.21064915  0.1325225   0.05
185484\n  0.01400669  0.02490351  0.02847982 -0.00291786]'
```

```
 '[0.13782788 0.07830002 0.0587317  0.05027437 0.29946197 0.04983954
\n 0.03984488 0.03958913 0.01976232 0.05865316]'
 '[ 0.25974505  0.0699356   0.02407657  0.22564459  0.30859688 -0.03
955727\n  0.04848697  0.          0.05497061  0.02745394]'
 '[ 0.06200502  0.06852263  0.12629086  0.15834307  0.33641488 -0.\n
0.11766042  0.08935594  0.0553327   0.03481868]'
 '[-0.01088757  0.07300643  0.17797892  0.          0.35717403  0.02
868402\n  0.05792379  0.12185049  0.05415508  0.1067278 ]'
 '[0.13513606 0.08243232 0.01268882 0.18798928 0.22489861 0.01811914
\n 0.0235254  0.02833165 0.04212795 0.        ]'
 '[0.39996332 0.04690359 0.01795939 0.01342455 0.07879634 0.06157932
\n 0.01091482 0.07649877 0.00917257 0.0790986 ]'
 '[0.03986424 0.02940483 0.02674348 0.23323632 0.25534202 0.05266985
\n 0.02055267 0.01294608 0.01053157 0.08804751]'
 '[3.26815182e-01 2.38763837e-02 1.75290224e-02 1.19161393e-01\n 1.1
9684198e-01 0.00000000e+00 2.16132348e-02 7.60072086e-02\n 4.1769312
5e-02 4.52451696e-05]'
 '[ 0.23296441  0.13672125 -0.02379449  0.13690477  0.28595006  0.03
707673\n  0.00776319  0.          0.03072477  0.0465746 ]'
 '[0.26782921 0.07054786 0.01629657 0.14863494 0.2455354  0.01575679
\n 0.00963322 0.         0.02878356 0.0054724 ]'
 '[ 0.32198492  0.00666568  0.04884657  0.          0.11477949  0.04
686317\n  0.02078401  0.10761295 -0.          0.01799848]'
 '[ 0.28983878  0.12967343  0.02164028  0.          0.312312   -0.\n
-0.          0.          0.10811877  0.07609464]'
 '[0.03600472 0.13570671 0.14253752 0.11290922 0.33832849 0.02591815
\n 0.02418937 0.06300623 0.01859214 0.0499253 ]'
 '[ 0.04930985  0.12370995  0.14408791  0.0118841   0.38827013  0.04
```

```
407355\n  0.04369237  0.06440638  0.0860947  -0.        ]'
 '[ 0.18594225  0.18054065 -0.01035722  0.20897919  0.15037876  0.02
249703\n  0.          0.00248287  0.06686776 -0.        ]'
 '[ 0.40459122  0.03747488  0.          0.13245572  0.17007618 -0.\n
-0.00085546  0.06107533  0.04114752  0.04490943]'
 '[ 0.22187451  0.05964707 -0.          0.0798845   0.38545173 -0.\n
0.00895587 -0.00335505  0.07718467  0.08431927]']
```

In [ ]: