用这些算法搞训练集和测试集

- 训练集不分割
- 测试集：不分割，男性，女性

In [1]:
```
import pandas as pd
import numpy as np
from glob import glob
from time import time

from surprise import Reader
from surprise import Dataset
from surprise.model_selection import cross_validate
from surprise import NormalPredictor
from surprise import KNNBasic
from surprise import KNNWithMeans
from surprise import KNNWithZScore
from surprise import KNNBaseline
from surprise import SVD
from surprise import BaselineOnly
from surprise import SVDpp
from surprise import NMF
from surprise import SlopeOne
from surprise import CoClustering
from surprise.accuracy import rmse, mae
from surprise import accuracy
from surprise.model_selection import train_test_split
from surprise.model_selection import GridSearchCV

from plotly.offline import init_notebook_mode, plot, iplot
import plotly.graph_objs as go
init_notebook_mode(connected=True)
```

In [4]:
```
def build_train_test_mf(df_train, df_test):
    reader = Reader(rating_scale=(1, 5))
    data_train = Dataset.load_from_df(df_train[['user_id', 'movie_id', 'rating']], reader)
    data_train = data_train.build_full_trainset()
    data_test = Dataset.load_from_df(df_test[['user_id', 'movie_id', 'rating']], reader)
    data_test = data_test.build_full_trainset().build_testset()

    df_test_m = df_test[df_test['sex'] == 'M']
    df_test_f = df_test[df_test['sex'] == 'F']
    data_test_m = Dataset.load_from_df(df_test_m[['user_id', 'movie_id', 'rating']], reader)
    data_test_m = data_test_m.build_full_trainset().build_testset()
    data_test_f = Dataset.load_from_df(df_test_f[['user_id', 'movie_id', 'rating']], reader)
    data_test_f = data_test_f.build_full_trainset().build_testset()

    return data_train, data_test, data_test_m, data_test_f
```

In [5]:
```
algorithms = {'SVD':SVD(), 'SVDpp':SVDpp(), 'SlopeOne':SlopeOne(), 'NMF':NMF(), 'NormalPredictor':NormalPredi
ctor(),
              'KNNBaseline':KNNBaseline(), 'KNNBasic':KNNBasic(), 'KNNWithMeans':KNNWithMeans(),
              'KNNWithZScore':KNNWithZScore(), 'BaselineOnly':BaselineOnly(), 'CoClustering':CoClustering()}
```

In [7]:
```
def train_single_algorithm_mf(algorithm_name, data_train, data_test, data_test_m, data_test_f, save_model=Fal
se):
    algorithms = {'SVD':SVD(), 'SVDpp':SVDpp(), 'SlopeOne':SlopeOne(), 'NMF':NMF(), 'NormalPredictor':NormalP
redictor(),
                  'KNNBaseline':KNNBaseline(), 'KNNBasic':KNNBasic(), 'KNNWithMeans':KNNWithMeans(),
                  'KNNWithZScore':KNNWithZScore(), 'BaselineOnly':BaselineOnly(), 'CoClustering':CoClustering()}
    assert(algorithm_name in algorithms), "{} does not exist!".format(algorithm_name)
    algo = algorithms[algorithm_name]
    start_time = time()
    algo.fit(data_train)
    # test
    predictions = algo.test(data_test)
    result = {}
    result['rmse'] = accuracy.rmse(predictions, verbose=True)
    result['mae'] = accuracy.mae(predictions, verbose=True)

    # test_m
    predictions_m = algo.test(data_test_m)
    result['rmse_m'] = accuracy.rmse(predictions_m, verbose=True)
    result['mae_m'] = accuracy.mae(predictions_m, verbose=True)

    # test_f
    predictions_f = algo.test(data_test_f)
    result['rmse_f'] = accuracy.rmse(predictions_f, verbose=True)
    result['mae_f'] = accuracy.mae(predictions_f, verbose=True)

    if save_model:
        result['model'] = algo

    print_result = "{:<20}|{:.2f} mins|rmse: {:.4f}|rmse_m: {:.4f}|rmse_f: {:.4f}|mae: {:.4f}|mae_m: {:.4f}|m
ae_f: {:.4f}"
    print_result = print_result.format(algorithm_name, (time() - start_time) / 60.,
                                       result['rmse'], result['rmse_m'], result['rmse_f'],
                                       result['mae'],result['mae_m'],result['mae_f'])
    print(print_result)
    return result
```

In [8]:
```
def get_mean_results(algorithms, all_results_list):
    for curr_algo_name in algorithms.keys():
        curr_algo_rmse = []
        curr_algo_mae = []
        for curr_all_results in all_results_list:
            curr_algo_rmse.append(curr_all_results[curr_algo_name]['rmse'])
            curr_algo_mae.append(curr_all_results[curr_algo_name]['mae'])
        print("{:<15}|rmse: {:.4f}+-{:.4f}|mae: {:.4f}+-{:.4f}".format(curr_algo_name,
                                                                   np.mean(curr_algo_rmse), np.std(curr_algo_
rmse),
                                                                   np.mean(curr_algo_mae), np.std(curr_algo_m
ae),
                                                                  ))
```

In [9]:
```
def get_mean_results_m(algorithms, all_results_list):
    for curr_algo_name in algorithms.keys():
        curr_algo_rmse = []
        curr_algo_mae = []
        for curr_all_results in all_results_list:
            curr_algo_rmse.append(curr_all_results[curr_algo_name]['rmse_m'])
            curr_algo_mae.append(curr_all_results[curr_algo_name]['mae_m'])
        print("{:<15}|rmse: {:.4f}+-{:.4f}|mae: {:.4f}+-{:.4f}".format(curr_algo_name,
                                                                   np.mean(curr_algo_rmse), np.std(curr_algo_
rmse),
                                                                   np.mean(curr_algo_mae), np.std(curr_algo_m
ae),
                                                                  ))
```

In [10]:
```
def get_mean_results_f(algorithms, all_results_list):
    for curr_algo_name in algorithms.keys():
        curr_algo_rmse = []
        curr_algo_mae = []
        for curr_all_results in all_results_list:
            curr_algo_rmse.append(curr_all_results[curr_algo_name]['rmse_f'])
            curr_algo_mae.append(curr_all_results[curr_algo_name]['mae_f'])
        print("{:<15}|rmse: {:.4f}+-{:.4f}|mae: {:.4f}+-{:.4f}".format(curr_algo_name,
                                                                   np.mean(curr_algo_rmse), np.std(curr_algo_
rmse),
                                                                   np.mean(curr_algo_mae), np.std(curr_algo_m
ae),
                                                                  ))
```

u1

In [11]:
```
# load
df_train = pd.read_csv("data/ml-100k_merged/u1.base")
df_test = pd.read_csv("data/ml-100k_merged/u1.test")
df_test.head(3)
```

Out[11]:

	movie_id	movie_title	user_id	age	sex	occupation	rating
0	1	Toy Story (1995)	5	33	F	other	4
1	2	GoldenEye (1995)	5	33	F	other	3
2	17	From Dusk Till Dawn (1996)	5	33	F	other	4

In [12]:
```
data_train, data_test, data_test_m, data_test_f = build_train_test_mf(df_train, df_test)
```

In [14]:
```
# start
all_results = {}
save_model = False
for algorithm_name in algorithms.keys():
    result = train_single_algorithm_mf(algorithm_name, data_train, data_test, data_test_m, data_test_f , save
_model)
    all_results[algorithm_name] = result
    print("===== ===== ===== =====")
```

```
RMSE: 0.9501
MAE:  0.7476
RMSE: 0.9327
MAE:  0.7334
RMSE: 1.0030
MAE:  0.7923
SVD                |0.07 mins|rmse: 0.9501|rmse_m: 0.9327|rmse_f: 1.0030|mae: 0.7476|mae_m: 0.7334|mae_f:
0.7923
===== ===== ===== =====
RMSE: 0.9333
MAE:  0.7322
RMSE: 0.9171
MAE:  0.7188
RMSE: 0.9824
MAE:  0.7744
SVDpp              |2.41 mins|rmse: 0.9333|rmse_m: 0.9171|rmse_f: 0.9824|mae: 0.7322|mae_m: 0.7188|mae_f:
0.7744
===== ===== ===== =====
RMSE: 0.9567
MAE:  0.7506
RMSE: 0.9364
MAE:  0.7353
RMSE: 1.0178
MAE:  0.7984
SlopeOne           |0.05 mins|rmse: 0.9567|rmse_m: 0.9364|rmse_f: 1.0178|mae: 0.7506|mae_m: 0.7353|mae_f:
0.7984
===== ===== ===== =====
RMSE: 0.9761
MAE:  0.7666
RMSE: 0.9564
MAE:  0.7511
RMSE: 1.0354
MAE:  0.8151
NMF                |0.06 mins|rmse: 0.9761|rmse_m: 0.9564|rmse_f: 1.0354|mae: 0.7666|mae_m: 0.7511|mae_f:
0.8151
===== ===== ===== =====
RMSE: 1.5405
MAE:  1.2338
RMSE: 1.5214
MAE:  1.2205
RMSE: 1.5937
MAE:  1.2862
NormalPredictor    |0.01 mins|rmse: 1.5405|rmse_m: 1.5214|rmse_f: 1.5937|mae: 1.2338|mae_m: 1.2205|mae_f:
1.2862
===== ===== ===== =====
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9418
MAE:  0.7413
RMSE: 0.9239
MAE:  0.7268
RMSE: 0.9958
MAE:  0.7870
KNNBaseline        |0.11 mins|rmse: 0.9418|rmse_m: 0.9239|rmse_f: 0.9958|mae: 0.7413|mae_m: 0.7268|mae_f:
0.7870
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9888
MAE:  0.7833
RMSE: 0.9653
MAE:  0.7640
RMSE: 1.0593
MAE:  0.8436
KNNBasic           |0.09 mins|rmse: 0.9888|rmse_m: 0.9653|rmse_f: 1.0593|mae: 0.7833|mae_m: 0.7640|mae_f:
0.8436
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9648
MAE:  0.7593
RMSE: 0.9487
MAE:  0.7462
RMSE: 1.0138
MAE:  0.8004
KNNWithMeans       |0.09 mins|rmse: 0.9648|rmse_m: 0.9487|rmse_f: 1.0138|mae: 0.7593|mae_m: 0.7462|mae_f:
0.8004
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9635
MAE:  0.7553
RMSE: 0.9480
MAE:  0.7432
RMSE: 1.0105
MAE:  0.7930
KNNWithZScore      |0.10 mins|rmse: 0.9635|rmse_m: 0.9480|rmse_f: 1.0105|mae: 0.7553|mae_m: 0.7432|mae_f:
```

```
0.7930
===== ===== ===== =====
Estimating biases using als...
RMSE: 0.9599
MAE:  0.7616
RMSE: 0.9402
MAE:  0.7452
RMSE: 1.0194
MAE:  0.8128
BaselineOnly        |0.01 mins|rmse: 0.9599|rmse_m: 0.9402|rmse_f: 1.0194|mae: 0.7616|mae_m: 0.7452|mae_f:
0.8128
===== ===== ===== =====
RMSE: 0.9840
MAE:  0.7710
RMSE: 0.9678
MAE:  0.7583
RMSE: 1.0332
MAE:  0.8108
CoClustering        |0.02 mins|rmse: 0.9840|rmse_m: 0.9678|rmse_f: 1.0332|mae: 0.7710|mae_m: 0.7583|mae_f:
0.8108
===== ===== ===== =====
In [15]: all_results
```

```
Out[15]: {'SVD': {'rmse': 0.9501496575462209,
           'mae': 0.7476167386044154,
           'rmse_m': 0.9326767407332749,
           'mae_m': 0.7333672898741278,
           'rmse_f': 1.0030089170720147,
           'mae_f': 0.7923345927099963},
          'SVDpp': {'rmse': 0.933338040162372,
           'mae': 0.732202621710163,
           'rmse_m': 0.9171453081402946,
           'mae_m': 0.7187540334913575,
           'rmse_f': 0.982422826410192,
           'mae_f': 0.7744072022014982},
          'SlopeOne': {'rmse': 0.9567192117629564,
           'mae': 0.7505898912181515,
           'rmse_m': 0.9364305105207631,
           'mae_m': 0.735341106793973,
           'rmse_f': 1.017766735667451,
           'mae_f': 0.7984438770161061},
          'NMF': {'rmse': 0.9761150331759124,
           'mae': 0.7665795816490396,
           'rmse_m': 0.9564488364314556,
           'mae_m': 0.751117038234773,
           'rmse_f': 1.035409185047486,
           'mae_f': 0.8151043894214753},
          'NormalPredictor': {'rmse': 1.5404847038481086,
           'mae': 1.2337729686840668,
           'rmse_m': 1.5214112491574798,
           'mae_m': 1.2204557728989258,
           'rmse_f': 1.5936513927888751,
           'mae_f': 1.2861535031753446},
          'KNNBaseline': {'rmse': 0.9417830614393241,
           'mae': 0.741335988489349,
           'rmse_m': 0.9239046984320807,
           'mae_m': 0.7267867628304324,
           'rmse_f': 0.9958071771760424,
           'mae_f': 0.7869946074773041},
          'KNNBasic': {'rmse': 0.9887958704696975,
           'mae': 0.7832791234223664,
           'rmse_m': 0.9652592879049,
           'mae_m': 0.7640470750729358,
           'rmse_f': 1.0592683488298327,
           'mae_f': 0.8436334535104719},
          'KNNWithMeans': {'rmse': 0.9648479897763116,
           'mae': 0.7592897649678887,
           'rmse_m': 0.9487244073199541,
           'mae_m': 0.7462028376783546,
           'rmse_f': 1.0137835324287456,
           'mae_f': 0.8003593751893583},
          'KNNWithZScore': {'rmse': 0.9634959916083691,
           'mae': 0.7552704605650169,
           'rmse_m': 0.94803035202629,
           'mae_m': 0.7432326224049755,
           'rmse_f': 1.0104947043217072,
           'mae_f': 0.7930478020451227},
          'BaselineOnly': {'rmse': 0.9599438333077737,
           'mae': 0.7615833440531363,
           'rmse_m': 0.9402121984604461,
           'mae_m': 0.7452492222584491,
           'rmse_f': 1.019388979805612,
           'mae_f': 0.8128433534179244},
          'CoClustering': {'rmse': 0.9839853680984596,
           'mae': 0.7710198656756778,
           'rmse_m': 0.9677725613560236,
           'mae_m': 0.7583385709681466,
           'rmse_f': 1.0332140508972687,
           'mae_f': 0.8108165130642826}}
```

u2

In [17]:
```
# load
df_train = pd.read_csv("data/ml-100k_merged/u2.base")
df_test = pd.read_csv("data/ml-100k_merged/u2.test")
data_train, data_test, data_test_m, data_test_f = build_train_test_mf(df_train, df_test)
all_results2 = {}
save_model = False
for algorithm_name in algorithms.keys():
    result = train_single_algorithm_mf(algorithm_name, data_train, data_test, data_test_m, data_test_f , save
_model)
    all_results2[algorithm_name] = result
    print("===== ===== ===== =====")
```

```
RMSE: 0.9398
MAE:  0.7387
RMSE: 0.9249
MAE:  0.7271
RMSE: 0.9827
MAE:  0.7735
SVD                 |0.07 mins|rmse: 0.9398|rmse_m: 0.9249|rmse_f: 0.9827|mae: 0.7387|mae_m: 0.7271|mae_f:
0.7735
===== ===== ===== =====
RMSE: 0.9206
MAE:  0.7214
RMSE: 0.9065
MAE:  0.7120
RMSE: 0.9617
MAE:  0.7496
SVDpp               |2.24 mins|rmse: 0.9206|rmse_m: 0.9065|rmse_f: 0.9617|mae: 0.7214|mae_m: 0.7120|mae_f:
0.7496
===== ===== ===== =====
RMSE: 0.9485
MAE:  0.7423
RMSE: 0.9291
MAE:  0.7290
RMSE: 1.0042
MAE:  0.7818
SlopeOne            |0.06 mins|rmse: 0.9485|rmse_m: 0.9291|rmse_f: 1.0042|mae: 0.7423|mae_m: 0.7290|mae_f:
0.7818
===== ===== ===== =====
RMSE: 0.9672
MAE:  0.7587
RMSE: 0.9485
MAE:  0.7450
RMSE: 1.0207
MAE:  0.7993
NMF                 |0.06 mins|rmse: 0.9672|rmse_m: 0.9485|rmse_f: 1.0207|mae: 0.7587|mae_m: 0.7450|mae_f:
0.7993
===== ===== ===== =====
RMSE: 1.5208
MAE:  1.2196
RMSE: 1.5143
MAE:  1.2198
RMSE: 1.5740
MAE:  1.2648
NormalPredictor     |0.00 mins|rmse: 1.5208|rmse_m: 1.5143|rmse_f: 1.5740|mae: 1.2196|mae_m: 1.2198|mae_f:
1.2648
===== ===== ===== =====
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9346
MAE:  0.7326
RMSE: 0.9180
MAE:  0.7207
RMSE: 0.9823
MAE:  0.7681
KNNBaseline         |0.10 mins|rmse: 0.9346|rmse_m: 0.9180|rmse_f: 0.9823|mae: 0.7326|mae_m: 0.7207|mae_f:
0.7681
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9848
MAE:  0.7750
RMSE: 0.9553
MAE:  0.7518
RMSE: 1.0677
MAE:  0.8441
KNNBasic            |0.08 mins|rmse: 0.9848|rmse_m: 0.9553|rmse_f: 1.0677|mae: 0.7750|mae_m: 0.7518|mae_f:
0.8441
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9571
MAE:  0.7510
RMSE: 0.9413
MAE:  0.7405
RMSE: 1.0026
MAE:  0.7824
KNNWithMeans        |0.09 mins|rmse: 0.9571|rmse_m: 0.9413|rmse_f: 1.0026|mae: 0.7510|mae_m: 0.7405|mae_f:
0.7824
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9576
MAE:  0.7480
RMSE: 0.9405
MAE:  0.7375
RMSE: 1.0068
MAE:  0.7794
KNNWithZScore       |0.09 mins|rmse: 0.9576|rmse_m: 0.9405|rmse_f: 1.0068|mae: 0.7480|mae_m: 0.7375|mae_f:
```

```
0.7794
===== ===== ===== =====
Estimating biases using als...
RMSE: 0.9477
MAE:  0.7494
RMSE: 0.9317
MAE:  0.7374
RMSE: 0.9935
MAE:  0.7852
BaselineOnly         |0.01 mins|rmse: 0.9477|rmse_m: 0.9317|rmse_f: 0.9935|mae: 0.7494|mae_m: 0.7374|mae_f: 
0.7852
===== ===== ===== =====
RMSE: 0.9671
MAE:  0.7549
RMSE: 0.9515
MAE:  0.7442
RMSE: 1.0123
MAE:  0.7868
CoClustering         |0.02 mins|rmse: 0.9671|rmse_m: 0.9515|rmse_f: 1.0123|mae: 0.7549|mae_m: 0.7442|mae_f: 
0.7868
===== ===== ===== =====
In [18]: all_results2
```

```
Out[18]: {'SVD': {'rmse': 0.9397627413299142,
  'mae': 0.7387300184664143,
  'rmse_m': 0.9248832259705064,
  'mae_m': 0.7270662956818287,
  'rmse_f': 0.9827476894452181,
  'mae_f': 0.7734706035576417},
 'SVDpp': {'rmse': 0.9206437469082495,
  'mae': 0.7214184754010948,
  'rmse_m': 0.9064505983700035,
  'mae_m': 0.7119503156710235,
  'rmse_f': 0.96167793233207,
  'mae_f': 0.7496195407757433},
 'SlopeOne': {'rmse': 0.948530796138768,
  'mae': 0.7422881947846279,
  'rmse_m': 0.9291061299296394,
  'mae_m': 0.7290161628638551,
  'rmse_f': 1.0041635866790495,
  'mae_f': 0.7818191543926908},
 'NMF': {'rmse': 0.9671620900792911,
  'mae': 0.758664037995159,
  'rmse_m': 0.9485273454143309,
  'mae_m': 0.7450161116000871,
  'rmse_f': 1.0206521030017706,
  'mae_f': 0.799314605314318},
 'NormalPredictor': {'rmse': 1.520844784735969,
  'mae': 1.2196190080691756,
  'rmse_m': 1.5142952373845495,
  'mae_m': 1.2197913092809822,
  'rmse_f': 1.5739755832595392,
  'mae_f': 1.2648277324059247},
 'KNNBaseline': {'rmse': 0.9345585443837379,
  'mae': 0.7326249556089116,
  'rmse_m': 0.9179725815200903,
  'mae_m': 0.7207024119018547,
  'rmse_f': 0.9823021902747693,
  'mae_f': 0.7681364429623555},
 'KNNBasic': {'rmse': 0.9847974058490248,
  'mae': 0.7750209854439283,
  'rmse_m': 0.9553458563677333,
  'mae_m': 0.7518230126377468,
  'rmse_f': 1.0677167258237217,
  'mae_f': 0.8441165189285028},
 'KNNWithMeans': {'rmse': 0.9570797740894271,
  'mae': 0.7510259453640877,
  'rmse_m': 0.9413071600677412,
  'mae_m': 0.7404886466669763,
  'rmse_f': 1.0025896380430122,
  'mae_f': 0.7824114582727513},
 'KNNWithZScore': {'rmse': 0.9575974988585253,
  'mae': 0.7480474779974006,
  'rmse_m': 0.9404902215345975,
  'mae_m': 0.7375336190590934,
  'rmse_f': 1.0068310185532539,
  'mae_f': 0.779363175209112},
 'BaselineOnly': {'rmse': 0.9476515797376743,
  'mae': 0.7493986092441747,
  'rmse_m': 0.9317431445873432,
  'mae_m': 0.737382006907687,
  'rmse_f': 0.9935269853966159,
  'mae_f': 0.7851902517315885},
 'CoClustering': {'rmse': 0.9671491768350303,
  'mae': 0.754942393014174,
  'rmse_m': 0.9515148490224595,
  'mae_m': 0.7442399970313226,
  'rmse_f': 1.0122865906736926,
  'mae_f': 0.7868196508322035}}
```

u3

In [19]:
```
# load
df_train = pd.read_csv("data/ml-100k_merged/u3.base")
df_test = pd.read_csv("data/ml-100k_merged/u3.test")
data_train, data_test, data_test_m, data_test_f = build_train_test_mf(df_train, df_test)
all_results3 = {}
save_model = False
for algorithm_name in algorithms.keys():
    result = train_single_algorithm_mf(algorithm_name, data_train, data_test, data_test_m, data_test_f , save
_model)
    all_results3[algorithm_name] = result
    print("===== ===== ===== =====")
```

```
RMSE: 0.9342
MAE:  0.7359
RMSE: 0.9176
MAE:  0.7229
RMSE: 0.9769
MAE:  0.7705
SVD                 |0.06 mins|rmse: 0.9342|rmse_m: 0.9176|rmse_f: 0.9769|mae: 0.7359|mae_m: 0.7229|mae_f:
0.7705
===== ===== ===== =====
RMSE: 0.9173
MAE:  0.7188
RMSE: 0.9017
MAE:  0.7079
RMSE: 0.9573
MAE:  0.7477
SVDpp               |2.18 mins|rmse: 0.9173|rmse_m: 0.9017|rmse_f: 0.9573|mae: 0.7188|mae_m: 0.7079|mae_f:
0.7477
===== ===== ===== =====
RMSE: 0.9457
MAE:  0.7427
RMSE: 0.9245
MAE:  0.7257
RMSE: 0.9999
MAE:  0.7877
SlopeOne            |0.06 mins|rmse: 0.9457|rmse_m: 0.9245|rmse_f: 0.9999|mae: 0.7427|mae_m: 0.7257|mae_f:
0.7877
===== ===== ===== =====
RMSE: 0.9564
MAE:  0.7526
RMSE: 0.9348
MAE:  0.7350
RMSE: 1.0116
MAE:  0.7994
NMF                 |0.06 mins|rmse: 0.9564|rmse_m: 0.9348|rmse_f: 1.0116|mae: 0.7526|mae_m: 0.7350|mae_f:
0.7994
===== ===== ===== =====
RMSE: 1.5092
MAE:  1.2101
RMSE: 1.5009
MAE:  1.2058
RMSE: 1.5563
MAE:  1.2486
NormalPredictor     |0.01 mins|rmse: 1.5092|rmse_m: 1.5009|rmse_f: 1.5563|mae: 1.2101|mae_m: 1.2058|mae_f:
1.2486
===== ===== ===== =====
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9292
MAE:  0.7317
RMSE: 0.9098
MAE:  0.7155
RMSE: 0.9788
MAE:  0.7745
KNNBaseline         |0.10 mins|rmse: 0.9292|rmse_m: 0.9098|rmse_f: 0.9788|mae: 0.7317|mae_m: 0.7155|mae_f:
0.7745
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9771
MAE:  0.7709
RMSE: 0.9515
MAE:  0.7497
RMSE: 1.0420
MAE:  0.8271
KNNBasic            |0.08 mins|rmse: 0.9771|rmse_m: 0.9515|rmse_f: 1.0420|mae: 0.7709|mae_m: 0.7497|mae_f:
0.8271
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9499
MAE:  0.7473
RMSE: 0.9323
MAE:  0.7333
RMSE: 0.9950
MAE:  0.7846
KNNWithMeans        |0.08 mins|rmse: 0.9499|rmse_m: 0.9323|rmse_f: 0.9950|mae: 0.7473|mae_m: 0.7333|mae_f:
0.7846
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9501
MAE:  0.7444
RMSE: 0.9324
MAE:  0.7308
RMSE: 0.9956
MAE:  0.7805
KNNWithZScore       |0.09 mins|rmse: 0.9501|rmse_m: 0.9324|rmse_f: 0.9956|mae: 0.7444|mae_m: 0.7308|mae_f:
```

```
0.7805
===== ===== ===== =====
Estimating biases using als...
RMSE: 0.9405
MAE:  0.7445
RMSE: 0.9239
MAE:  0.7304
RMSE: 0.9834
MAE:  0.7819
BaselineOnly        |0.01 mins|rmse: 0.9405|rmse_m: 0.9239|rmse_f: 0.9834|mae: 0.7445|mae_m: 0.7304|mae_f:
0.7819
===== ===== ===== =====
RMSE: 0.9631
MAE:  0.7537
RMSE: 0.9457
MAE:  0.7401
RMSE: 1.0079
MAE:  0.7899
CoClustering        |0.02 mins|rmse: 0.9631|rmse_m: 0.9457|rmse_f: 1.0079|mae: 0.7537|mae_m: 0.7401|mae_f:
0.7899
===== ===== ===== =====
In [20]: all_results3
```

```
Out[20]: {'SVD': {'rmse': 0.9341918164958688,
           'mae': 0.7358958337016195,
           'rmse_m': 0.9176222871100206,
           'mae_m': 0.7228684385035556,
           'rmse_f': 0.9768521706892004,
           'mae_f': 0.7705093059311069},
          'SVDpp': {'rmse': 0.9172556112125749,
           'mae': 0.7187735718234424,
           'rmse_m': 0.9017375688681257,
           'mae_m': 0.7078821847327575,
           'rmse_f': 0.9572653108201846,
           'mae_f': 0.7477117224569662},
          'SlopeOne': {'rmse': 0.9457210548243965,
           'mae': 0.7426791889651031,
           'rmse_m': 0.9244926340305414,
           'mae_m': 0.7257223874262158,
           'rmse_f': 0.9999373581366584,
           'mae_f': 0.787732998283365},
          'NMF': {'rmse': 0.9563795067096175,
           'mae': 0.7526387952431609,
           'rmse_m': 0.9347506464258216,
           'mae_m': 0.7350489064157646,
           'rmse_f': 1.0116026938470513,
           'mae_f': 0.7993747020910117},
          'NormalPredictor': {'rmse': 1.5092316820750777,
           'mae': 1.2101008633966046,
           'rmse_m': 1.5008794927298108,
           'mae_m': 1.20583659666015,
           'rmse_f': 1.5563291725995116,
           'mae_f': 1.248552300229709},
          'KNNBaseline': {'rmse': 0.9291944905970315,
           'mae': 0.7316578581069132,
           'rmse_m': 0.9098094874331983,
           'mae_m': 0.7155300834101734,
           'rmse_f': 0.9788366501188447,
           'mae_f': 0.7745089632665998},
          'KNNBasic': {'rmse': 0.97709795253605,
           'mae': 0.7708818274785253,
           'rmse_m': 0.9515273870845976,
           'mae_m': 0.7497265778168632,
           'rmse_f': 1.0419942078002027,
           'mae_f': 0.8270908113574086},
          'KNNWithMeans': {'rmse': 0.9498546994004081,
           'mae': 0.7473468557726317,
           'rmse_m': 0.9322960708204558,
           'mae_m': 0.7333428249497211,
           'rmse_f': 0.9950033396412336,
           'mae_f': 0.7845552251066443},
          'KNNWithZScore': {'rmse': 0.9501250111896186,
           'mae': 0.7444131337643888,
           'rmse_m': 0.9324382318518022,
           'mae_m': 0.7308229911207497,
           'rmse_f': 0.9955930274427556,
           'mae_f': 0.7805218122713777},
          'BaselineOnly': {'rmse': 0.9405230282786979,
           'mae': 0.7445158474930392,
           'rmse_m': 0.9238773480739032,
           'mae_m': 0.7304425721174526,
           'rmse_f': 0.9833822829673046,
           'mae_f': 0.7819081979195617},
          'CoClustering': {'rmse': 0.9630848185609732,
           'mae': 0.7537123124860914,
           'rmse_m': 0.9456558907969556,
           'mae_m': 0.740107721381658,
           'rmse_f': 1.0079299505903292,
           'mae_f': 0.7898593802020397}}
```

u4

In [21]:
```
# load
df_train = pd.read_csv("data/ml-100k_merged/u4.base")
df_test = pd.read_csv("data/ml-100k_merged/u4.test")
data_train, data_test, data_test_m, data_test_f = build_train_test_mf(df_train, df_test)
all_results4 = {}
save_model = False
for algorithm_name in algorithms.keys():
    result = train_single_algorithm_mf(algorithm_name, data_train, data_test, data_test_m, data_test_f , save
_model)
    all_results4[algorithm_name] = result
    print("===== ===== ===== =====")
```

```
RMSE: 0.9327
MAE:  0.7349
RMSE: 0.9138
MAE:  0.7201
RMSE: 0.9842
MAE:  0.7768
SVD                 |0.07 mins|rmse: 0.9327|rmse_m: 0.9138|rmse_f: 0.9842|mae: 0.7349|mae_m: 0.7201|mae_f:
0.7768
===== ===== ===== =====
RMSE: 0.9174
MAE:  0.7207
RMSE: 0.8995
MAE:  0.7071
RMSE: 0.9663
MAE:  0.7590
SVDpp               |2.23 mins|rmse: 0.9174|rmse_m: 0.8995|rmse_f: 0.9663|mae: 0.7207|mae_m: 0.7071|mae_f:
0.7590
===== ===== ===== =====
RMSE: 0.9432
MAE:  0.7402
RMSE: 0.9186
MAE:  0.7221
RMSE: 1.0095
MAE:  0.7915
SlopeOne            |0.06 mins|rmse: 0.9432|rmse_m: 0.9186|rmse_f: 1.0095|mae: 0.7402|mae_m: 0.7221|mae_f:
0.7915
===== ===== ===== =====
RMSE: 0.9640
MAE:  0.7581
RMSE: 0.9389
MAE:  0.7390
RMSE: 1.0319
MAE:  0.8124
NMF                 |0.06 mins|rmse: 0.9640|rmse_m: 0.9389|rmse_f: 1.0319|mae: 0.7581|mae_m: 0.7390|mae_f:
0.8124
===== ===== ===== =====
RMSE: 1.5148
MAE:  1.2151
RMSE: 1.4983
MAE:  1.2036
RMSE: 1.5587
MAE:  1.2444
NormalPredictor     |0.01 mins|rmse: 1.5148|rmse_m: 1.4983|rmse_f: 1.5587|mae: 1.2151|mae_m: 1.2036|mae_f:
1.2444
===== ===== ===== =====
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9260
MAE:  0.7301
RMSE: 0.9030
MAE:  0.7124
RMSE: 0.9884
MAE:  0.7803
KNNBaseline         |0.10 mins|rmse: 0.9260|rmse_m: 0.9030|rmse_f: 0.9884|mae: 0.7301|mae_m: 0.7124|mae_f:
0.7803
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9704
MAE:  0.7671
RMSE: 0.9422
MAE:  0.7445
RMSE: 1.0463
MAE:  0.8313
KNNBasic            |0.08 mins|rmse: 0.9704|rmse_m: 0.9422|rmse_f: 1.0463|mae: 0.7671|mae_m: 0.7445|mae_f:
0.8313
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9448
MAE:  0.7444
RMSE: 0.9250
MAE:  0.7284
RMSE: 0.9988
MAE:  0.7899
KNNWithMeans        |0.09 mins|rmse: 0.9448|rmse_m: 0.9250|rmse_f: 0.9988|mae: 0.7444|mae_m: 0.7284|mae_f:
0.7899
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9452
MAE:  0.7413
RMSE: 0.9245
MAE:  0.7251
RMSE: 1.0016
MAE:  0.7870
KNNWithZScore       |0.09 mins|rmse: 0.9452|rmse_m: 0.9245|rmse_f: 1.0016|mae: 0.7413|mae_m: 0.7251|mae_f:
```

```
0.7870
===== ===== ===== =====
Estimating biases using als...
RMSE: 0.9383
MAE:  0.7442
RMSE: 0.9183
MAE:  0.7293
RMSE: 0.9928
MAE:  0.7864
BaselineOnly        |0.00 mins|rmse: 0.9383|rmse_m: 0.9183|rmse_f: 0.9928|mae: 0.7442|mae_m: 0.7293|mae_f:
0.7864
===== ===== ===== =====
RMSE: 0.9600
MAE:  0.7508
RMSE: 0.9394
MAE:  0.7352
RMSE: 1.0160
MAE:  0.7948
CoClustering        |0.02 mins|rmse: 0.9600|rmse_m: 0.9394|rmse_f: 1.0160|mae: 0.7508|mae_m: 0.7352|mae_f:
0.7948
===== ===== ===== =====
In [22]: all_results4
```

```
Out[22]: {'SVD': {'rmse': 0.9326891300457406,
            'mae': 0.7349325099973466,
            'rmse_m': 0.9137889739088161,
            'mae_m': 0.7201494645221024,
            'rmse_f': 0.9842106816964813,
            'mae_f': 0.776767792654022},
           'SVDpp': {'rmse': 0.9174445554301403,
            'mae': 0.7206841273991911,
            'rmse_m': 0.8995432826121089,
            'mae_m': 0.7071294588646992,
            'rmse_f': 0.9663087449974147,
            'mae_f': 0.7590431644736307},
           'SlopeOne': {'rmse': 0.9431506535440269,
            'mae': 0.7402308642861269,
            'rmse_m': 0.9185618954977249,
            'mae_m': 0.7221299611743005,
            'rmse_f': 1.0094950967797836,
            'mae_f': 0.7914555188603464},
           'NMF': {'rmse': 0.9640065599052885,
            'mae': 0.7581422258526658,
            'rmse_m': 0.9388572363579029,
            'mae_m': 0.7389731265873072,
            'rmse_f': 1.0318614254507883,
            'mae_f': 0.8123898223565857},
           'NormalPredictor': {'rmse': 1.5148012580629089,
            'mae': 1.215103128246933,
            'rmse_m': 1.4983350134651243,
            'mae_m': 1.2036223714640448,
            'rmse_f': 1.5587299774693848,
            'mae_f': 1.2444110628044935},
           'KNNBaseline': {'rmse': 0.9260155325581058,
            'mae': 0.7300987706659998,
            'rmse_m': 0.9029576148739519,
            'mae_m': 0.71236584642452,
            'rmse_f': 0.9883573799791182,
            'mae_f': 0.7802820633585676},
           'KNNBasic': {'rmse': 0.9704489972276842,
            'mae': 0.7671418382414986,
            'rmse_m': 0.9422009990745012,
            'mae_m': 0.7444565238916466,
            'rmse_f': 1.0462644580403377,
            'mae_f': 0.8313401483644617},
           'KNNWithMeans': {'rmse': 0.9448167988667769,
            'mae': 0.7444309286996698,
            'rmse_m': 0.9249965086431827,
            'mae_m': 0.7283759795607595,
            'rmse_f': 0.9987779654460475,
            'mae_f': 0.7898656353972604},
           'KNNWithZScore': {'rmse': 0.9452156164912281,
            'mae': 0.7412841328617357,
            'rmse_m': 0.9244566039804679,
            'mae_m': 0.7251239650862336,
            'rmse_f': 1.001633736912796,
            'mae_f': 0.787016603062113},
           'BaselineOnly': {'rmse': 0.938284026686687,
            'mae': 0.7442326440918581,
            'rmse_m': 0.9182580045610229,
            'mae_m': 0.729325255826984,
            'rmse_f': 0.9927698943810231,
            'mae_f': 0.7864198106522386},
           'CoClustering': {'rmse': 0.9599890835342897,
            'mae': 0.7507564478210741,
            'rmse_m': 0.9393897898127993,
            'mae_m': 0.7351920925550179,
            'rmse_f': 1.0160232702029648,
            'mae_f': 0.7948027982848387}}
```

u5

In [23]:
```
# load
df_train = pd.read_csv("data/ml-100k_merged/u5.base")
df_test = pd.read_csv("data/ml-100k_merged/u5.test")
data_train, data_test, data_test_m, data_test_f = build_train_test_mf(df_train, df_test)
all_results5 = {}
save_model = False
for algorithm_name in algorithms.keys():
    result = train_single_algorithm_mf(algorithm_name, data_train, data_test, data_test_m, data_test_f , save
_model)
    all_results5[algorithm_name] = result
    print("===== ===== ===== =====")
```

```
RMSE: 0.9342
MAE:  0.7389
RMSE: 0.9141
MAE:  0.7228
RMSE: 0.9893
MAE:  0.7851
SVD                 |0.06 mins|rmse: 0.9342|rmse_m: 0.9141|rmse_f: 0.9893|mae: 0.7389|mae_m: 0.7228|mae_f:
0.7851
===== ===== ===== =====
RMSE: 0.9171
MAE:  0.7256
RMSE: 0.8980
MAE:  0.7105
RMSE: 0.9696
MAE:  0.7687
SVDpp               |2.27 mins|rmse: 0.9171|rmse_m: 0.8980|rmse_f: 0.9696|mae: 0.7256|mae_m: 0.7105|mae_f:
0.7687
===== ===== ===== =====
RMSE: 0.9408
MAE:  0.7436
RMSE: 0.9165
MAE:  0.7244
RMSE: 1.0070
MAE:  0.7983
SlopeOne            |0.05 mins|rmse: 0.9408|rmse_m: 0.9165|rmse_f: 1.0070|mae: 0.7436|mae_m: 0.7244|mae_f:
0.7983
===== ===== ===== =====
RMSE: 0.9638
MAE:  0.7588
RMSE: 0.9413
MAE:  0.7417
RMSE: 1.0253
MAE:  0.8077
NMF                 |0.06 mins|rmse: 0.9638|rmse_m: 0.9413|rmse_f: 1.0253|mae: 0.7588|mae_m: 0.7417|mae_f:
0.8077
===== ===== ===== =====
RMSE: 1.5182
MAE:  1.2217
RMSE: 1.5070
MAE:  1.2098
RMSE: 1.5179
MAE:  1.2163
NormalPredictor     |0.00 mins|rmse: 1.5182|rmse_m: 1.5070|rmse_f: 1.5179|mae: 1.2217|mae_m: 1.2098|mae_f:
1.2163
===== ===== ===== =====
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9299
MAE:  0.7370
RMSE: 0.9079
MAE:  0.7189
RMSE: 0.9900
MAE:  0.7885
KNNBaseline         |0.10 mins|rmse: 0.9299|rmse_m: 0.9079|rmse_f: 0.9900|mae: 0.7370|mae_m: 0.7189|mae_f:
0.7885
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9792
MAE:  0.7756
RMSE: 0.9493
MAE:  0.7514
RMSE: 1.0602
MAE:  0.8445
KNNBasic            |0.08 mins|rmse: 0.9792|rmse_m: 0.9493|rmse_f: 1.0602|mae: 0.7756|mae_m: 0.7514|mae_f:
0.8445
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9471
MAE:  0.7505
RMSE: 0.9243
MAE:  0.7324
RMSE: 1.0093
MAE:  0.8022
KNNWithMeans        |0.09 mins|rmse: 0.9471|rmse_m: 0.9243|rmse_f: 1.0093|mae: 0.7505|mae_m: 0.7324|mae_f:
0.8022
===== ===== ===== =====
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9471
MAE:  0.7476
RMSE: 0.9237
MAE:  0.7294
RMSE: 1.0108
MAE:  0.7996
KNNWithZScore       |0.09 mins|rmse: 0.9471|rmse_m: 0.9237|rmse_f: 1.0108|mae: 0.7476|mae_m: 0.7294|mae_f:
```

```
0.7996
===== ===== ===== =====
Estimating biases using als...
RMSE: 0.9423
MAE:  0.7499
RMSE: 0.9227
MAE:  0.7336
RMSE: 0.9959
MAE:  0.7967
BaselineOnly        |0.00 mins|rmse: 0.9423|rmse_m: 0.9227|rmse_f: 0.9959|mae: 0.7499|mae_m: 0.7336|mae_f:
0.7967
===== ===== ===== =====
RMSE: 0.9560
MAE:  0.7523
RMSE: 0.9336
MAE:  0.7354
RMSE: 1.0172
MAE:  0.8004
CoClustering        |0.02 mins|rmse: 0.9560|rmse_m: 0.9336|rmse_f: 1.0172|mae: 0.7523|mae_m: 0.7354|mae_f:
0.8004
===== ===== ===== =====
In [24]: all_results5
```

```
Out[24]: {'SVD': {'rmse': 0.9341522051026263,
           'mae': 0.738940042216897,
           'rmse_m': 0.9140563897022423,
           'mae_m': 0.7227734677050554,
           'rmse_f': 0.9892688735166905,
           'mae_f': 0.7850844120559579},
          'SVDpp': {'rmse': 0.917125766367807,
           'mae': 0.7256035298339313,
           'rmse_m': 0.8980192604838565,
           'mae_m': 0.7104935713633109,
           'rmse_f': 0.9695927623545668,
           'mae_f': 0.7687319929112787},
          'SlopeOne': {'rmse': 0.9407889914875407,
           'mae': 0.7436115185387002,
           'rmse_m': 0.9164575173655867,
           'mae_m': 0.7244479766958435,
           'rmse_f': 1.0070098165777601,
           'mae_f': 0.7983101537737269},
          'NMF': {'rmse': 0.963794759206967,
           'mae': 0.7588251088812626,
           'rmse_m': 0.9413132771591501,
           'mae_m': 0.7416910316814658,
           'rmse_f': 1.025255734823719,
           'mae_f': 0.8077310285972373},
          'NormalPredictor': {'rmse': 1.5181892846532634,
           'mae': 1.2216992509074702,
           'rmse_m': 1.5070285139213548,
           'mae_m': 1.2098381401421603,
           'rmse_f': 1.5178577927459407,
           'mae_f': 1.2162516356280877},
          'KNNBaseline': {'rmse': 0.9299180021351924,
           'mae': 0.7369989764410888,
           'rmse_m': 0.9079231489916165,
           'mae_m': 0.7189422641928898,
           'rmse_f': 0.9900137787394035,
           'mae_f': 0.7885383800078795},
          'KNNBasic': {'rmse': 0.9792464045505916,
           'mae': 0.7755651888896984,
           'rmse_m': 0.9492587722193442,
           'mae_m': 0.7514051082353181,
           'rmse_f': 1.0601848084566308,
           'mae_f': 0.8445254807709903},
          'KNNWithMeans': {'rmse': 0.9470911255392472,
           'mae': 0.7505104648653849,
           'rmse_m': 0.924316333993949,
           'mae_m': 0.7323994720029184,
           'rmse_f': 1.0092743846563494,
           'mae_f': 0.8022048019796636},
          'KNNWithZScore': {'rmse': 0.9470581257878241,
           'mae': 0.7476369812311432,
           'rmse_m': 0.9236765362586835,
           'mae_m': 0.7294285308597599,
           'rmse_f': 1.0108257172858266,
           'mae_f': 0.7996094920136753},
          'BaselineOnly': {'rmse': 0.9422794835917605,
           'mae': 0.7499396915080997,
           'rmse_m': 0.9227453655031735,
           'mae_m': 0.733559636875125,
           'rmse_f': 0.995930543562663,
           'mae_f': 0.796693399191466},
          'CoClustering': {'rmse': 0.9559537782330744,
           'mae': 0.7522740539985507,
           'rmse_m': 0.933553956524812,
           'mae_m': 0.7354102161515943,
           'rmse_f': 1.0171799938873143,
           'mae_f': 0.8004086275871556}}
```

现在看下在这5个数据集下的综合水平

In [25]:
```
all_results_list = [all_results, all_results2, all_results3, all_results4, all_results5]
get_mean_results(algorithms, all_results_list)
```

```
SVD            |rmse: 0.9382+-0.0065|mae: 0.7392+-0.0045
SVDpp          |rmse: 0.9212+-0.0062|mae: 0.7237+-0.0048
SlopeOne       |rmse: 0.9470+-0.0055|mae: 0.7439+-0.0035
NMF            |rmse: 0.9655+-0.0064|mae: 0.7590+-0.0044
NormalPredictor|rmse: 1.5207+-0.0106|mae: 1.2201+-0.0079
KNNBaseline    |rmse: 0.9323+-0.0055|mae: 0.7345+-0.0041
KNNBasic       |rmse: 0.9801+-0.0063|mae: 0.7744+-0.0054
KNNWithMeans   |rmse: 0.9527+-0.0073|mae: 0.7505+-0.0050
KNNWithZScore  |rmse: 0.9527+-0.0069|mae: 0.7473+-0.0047
BaselineOnly   |rmse: 0.9457+-0.0077|mae: 0.7499+-0.0063
CoClustering   |rmse: 0.9660+-0.0097|mae: 0.7565+-0.0074
```

In [26]:
```
# 全训练集，男测试集
get_mean_results_m(algorithms, all_results_list)
```

```
SVD            |rmse: 0.9206+-0.0072|mae: 0.7252+-0.0046
SVDpp          |rmse: 0.9046+-0.0069|mae: 0.7112+-0.0041
SlopeOne       |rmse: 0.9250+-0.0072|mae: 0.7273+-0.0046
NMF            |rmse: 0.9440+-0.0077|mae: 0.7424+-0.0055
NormalPredictor|rmse: 1.5084+-0.0085|mae: 1.2119+-0.0070
KNNBaseline    |rmse: 0.9125+-0.0075|mae: 0.7189+-0.0049
KNNBasic       |rmse: 0.9527+-0.0076|mae: 0.7523+-0.0064
KNNWithMeans   |rmse: 0.9343+-0.0095|mae: 0.7362+-0.0064
KNNWithZScore  |rmse: 0.9338+-0.0094|mae: 0.7332+-0.0064
BaselineOnly   |rmse: 0.9274+-0.0078|mae: 0.7352+-0.0058
CoClustering   |rmse: 0.9476+-0.0118|mae: 0.7427+-0.0085
```

In [27]:
```
# 全训练集，女测试集
get_mean_results_f(algorithms, all_results_list)
```

```
SVD            |rmse: 0.9872+-0.0088|mae: 0.7796+-0.0080
SVDpp          |rmse: 0.9675+-0.0086|mae: 0.7599+-0.0104
SlopeOne       |rmse: 1.0077+-0.0060|mae: 0.7916+-0.0064
NMF            |rmse: 1.0250+-0.0084|mae: 0.8068+-0.0065
NormalPredictor|rmse: 1.5601+-0.0250|mae: 1.2520+-0.0231
KNNBaseline    |rmse: 0.9871+-0.0060|mae: 0.7797+-0.0076
KNNBasic       |rmse: 1.0551+-0.0095|mae: 0.8381+-0.0074
KNNWithMeans   |rmse: 1.0039+-0.0068|mae: 0.7919+-0.0081
KNNWithZScore  |rmse: 1.0051+-0.0058|mae: 0.7879+-0.0076
BaselineOnly   |rmse: 0.9970+-0.0120|mae: 0.7926+-0.0113
CoClustering   |rmse: 1.0173+-0.0086|mae: 0.7965+-0.0085
```