使用dl中使用的ml-1m数据集

- 分割训练集与测试集 --> 之后可以分别聚合rmse / recall
  - 男性
  - 女性
  - 7种年龄
- 计算rmse / recall

$$\text{RMSE}(X) = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}{n}}$$

$$\text{RMSE}(XY) = \sqrt{\frac{\text{RMSE}(X)^2 \times n_X + \text{RMSE}(Y)^2 \times n_Y}{n_X + n_Y}}$$

In [1]:

```python
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

import math
import copy
import pickle
from pathlib import Path
from itertools import zip_longest
from collections import defaultdict
```

1. 加载dl时使用的ml-1m数据集

In [2]:

```python
df = pd.read_csv("ml-1m_dl.csv")
datasets = pickle.load(open('ml-1m_dl.pkl','rb'))
# df_train
df_train = datasets['train'][0].copy()
df_train['rating'] = datasets['train'][1].astype(np.int64)
# df_test
df_test = datasets['val'][0].copy()
df_test['rating'] = datasets['val'][1].astype(np.int64)
```

In [3]:

```python
def build_train_dataset(reader, df_train):
    print(len(df_train))
    data_train = Dataset.load_from_df(df_train[['user_id', 'movie_id', 'rating']], reader)
    data_train = data_train.build_full_trainset()
    return data_train

def build_test_dataset(reader, df_test):
    print(len(df_test))
    data_test = Dataset.load_from_df(df_test[['user_id', 'movie_id', 'rating']], reader)
    data_test = data_test.build_full_trainset().build_testset()
    return data_test
```

In [4]:

```
df_train[df_train['sex_index'] == 1]
```

Out[4]:

	user_id	movie_id	sex_index	age_index	rating
341591	3600	609	1	3	4
470922	4889	1291	1	1	5
630004	5837	1573	1	2	4
947598	4428	1339	1	2	4
508884	5312	2202	1	2	3
...	...	...	...	...	...
836489	4078	482	1	2	3
491263	5086	608	1	2	5
791624	3457	1090	1	2	4
470924	4889	1302	1	1	5
128037	1317	1276	1	4	4

602878 rows × 5 columns

In [5]:

```
# 分割训练集测试集
reader = Reader(rating_scale=(1, 5))

# 性别
data_train_m = build_train_dataset(reader, df_train[df_train['sex_index']==1])
data_train_f = build_train_dataset(reader, df_train[df_train['sex_index'] == 0])
data_test_m = build_test_dataset(reader, df_test[df_test['sex_index'] == 1])
data_test_f = build_test_dataset(reader, df_test[df_test['sex_index'] == 0])
data_train_list = [data_train_m, data_train_f]
data_test_list = [data_test_m, data_test_f]
print("-----")

for age_index in range(7):
    df_train_age = df_train[df_train['age_index'] == age_index]
    df_test_age = df_test[df_test['age_index'] == age_index]
    print("AgeIndex {}: train {}, test {}".format(age_index, len(df_train_age), len(df_test_age)))
    data_train_age = build_train_dataset(reader, df_train_age)
    data_test_age = build_test_dataset(reader, df_test_age)
    data_train_list.append(data_train_age)
    data_test_list.append(data_test_age)
```

```
602878
197289
150891
49151
-----
AgeIndex 0: train 21843, test 5368
21843
5368
AgeIndex 1: train 146739, test 36797
146739
36797
AgeIndex 2: train 316256, test 79300
316256
79300
AgeIndex 3: train 159120, test 39883
159120
39883
AgeIndex 4: train 67096, test 16537
67096
16537
AgeIndex 5: train 58020, test 14470
58020
14470
AgeIndex 6: train 31093, test 7687
31093
7687
```

In [6]:

```
# 单个训练-测试集的结果 --> 注意记录训练样本和测试样本量
def train_single_algorithm(algorithm_name, data_train, data_test):
    algorithms = {'SVD':SVD(), 'SVDpp':SVDpp(), 'SlopeOne':SlopeOne(), 'NMF':NMF(), 'NormalPredictor':NormalP
redictor(),
              'KNNBaseline':KNNBaseline(), 'KNNBasic':KNNBasic(), 'KNNWithMeans':KNNWithMeans(),
              'KNNWithZScore':KNNWithZScore(), 'BaselineOnly':BaselineOnly(), 'CoClustering':CoClustering()}
    algo = algorithms[algorithm_name]
    start_time = time()
    print("Start training: {}".format(algorithm_name))
    algo.fit(data_train)

    # test
    # print("Start testing on full test set")
    predictions = algo.test(data_test)
    result = {}
    result['n_samples'] = len(predictions)
    result['rmse'] = accuracy.rmse(predictions, verbose=True)
    result['recall'] = precision_recall_at_k(predictions, k=10, threshold=3.5)[1]
    print("Algorithm {} finished with {:.2f} mins".format(algorithm_name, (time() - start_time) / 60.))
    return result
```

In [7]:

```
def show_single_result(algo_name, result):
    print("Algo: {}".format(algo_name))
    for key in sorted(result.keys(), reverse=True):
        print("{:<13}: {:.4f}".format(key, result[key]))
```

In [8]:

```
def precision_recall_at_k(predictions, k=10, threshold=3.5):
    '''Return precision and recall at k metrics for each user.'''
    # First map the predictions to each user.
    user_est_true = defaultdict(list)
    for uid, _, true_r, est, _ in predictions:
        user_est_true[uid].append((est, true_r))
    precisions = dict()
    recalls = dict()
    for uid, user_ratings in user_est_true.items():
        # Sort user ratings by estimated value
        user_ratings.sort(key=lambda x: x[0], reverse=True)
        # Number of relevant items
        n_rel = sum((true_r >= threshold) for (_, true_r) in user_ratings)
        # Number of recommended items in top k
        n_rec_k = sum((est >= threshold) for (est, _) in user_ratings[:k])
        # Number of relevant and recommended items in top k
        n_rel_and_rec_k = sum(((true_r >= threshold) and (est >= threshold))
                              for (est, true_r) in user_ratings[:k])
        # Precision@K: Proportion of recommended items that are relevant
        precisions[uid] = n_rel_and_rec_k / n_rec_k if n_rec_k != 0 else 1
        # Recall@K: Proportion of relevant items that are recommended
        recalls[uid] = n_rel_and_rec_k / n_rel if n_rel != 0 else 1

        precisions_mean = sum(prec for prec in precisions.values()) / len(precisions)
        recalls_mean = sum(rec for rec in recalls.values()) / len(recalls)
    return precisions_mean, recalls_mean
```

Now let's start

In [9]:

```
# 不使用SVDpp --> 太慢了
algorithms = {'SVD':SVD(), 'SlopeOne':SlopeOne(), 'NMF':NMF(), 'NormalPredictor':NormalPredictor(),
              'KNNBaseline':KNNBaseline(), 'KNNBasic':KNNBasic(), 'KNNWithMeans':KNNWithMeans(),
              'KNNWithZScore':KNNWithZScore(), 'BaselineOnly':BaselineOnly(), 'CoClustering':CoClustering()}
data_names = ['Male', 'Female', 'Age1', 'Age18', 'Age25', 'Age35', 'Age45', 'Age50', 'Age56']
```

In [10]:

```
all_results_all_dataset = {}
```

In [11]:

```
index = 0
data_name = data_names[index]
data_train = data_train_list[index]
data_test = data_test_list[index]
all_results = {}
print("Start training on dataset: {}".format(data_name))
for algorithm_name in algorithms.keys():
    result = train_single_algorithm(algorithm_name, data_train, data_test)
    all_results[algorithm_name] = result
    print("===== ===== ===== =====")
all_results_all_dataset[data_name] = all_results
```

```
Start training on dataset: Male
Start training: SVD
RMSE: 0.8666
Algorithm SVD finished with 0.64 mins
===== ===== ===== =====
Start training: SlopeOne
RMSE: 0.8953
Algorithm SlopeOne finished with 0.71 mins
===== ===== ===== =====
Start training: NMF
RMSE: 0.9057
Algorithm NMF finished with 0.62 mins
===== ===== ===== =====
Start training: NormalPredictor
RMSE: 1.5088
Algorithm NormalPredictor finished with 0.05 mins
===== ===== ===== =====
Start training: KNNBaseline
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.8834
Algorithm KNNBaseline finished with 1.24 mins
===== ===== ===== =====
Start training: KNNBasic
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9127
Algorithm KNNBasic finished with 1.15 mins
===== ===== ===== =====
Start training: KNNWithMeans
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9182
Algorithm KNNWithMeans finished with 1.18 mins
===== ===== ===== =====
Start training: KNNWithZScore
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9186
Algorithm KNNWithZScore finished with 1.22 mins
===== ===== ===== =====
Start training: BaselineOnly
Estimating biases using als...
RMSE: 0.8981
Algorithm BaselineOnly finished with 0.07 mins
===== ===== ===== =====
Start training: CoClustering
RMSE: 0.9065
Algorithm CoClustering finished with 0.23 mins
===== ===== ===== =====
```

In [12]:

```
index = 1
data_name = data_names[index]
data_train = data_train_list[index]
data_test = data_test_list[index]
all_results = {}
print("Start training on dataset: {}".format(data_name))
for algorithm_name in algorithms.keys():
    result = train_single_algorithm(algorithm_name, data_train, data_test)
    all_results[algorithm_name] = result
    print("===== ===== ===== =====")
all_results_all_dataset[data_name] = all_results
```

```
Start training on dataset: Female
Start training: SVD
RMSE: 0.9310
Algorithm SVD finished with 0.19 mins
===== ===== ===== =====
Start training: SlopeOne
RMSE: 0.9366
Algorithm SlopeOne finished with 0.18 mins
===== ===== ===== =====
Start training: NMF
RMSE: 0.9505
Algorithm NMF finished with 0.19 mins
===== ===== ===== =====
Start training: NormalPredictor
RMSE: 1.4906
Algorithm NormalPredictor finished with 0.01 mins
===== ===== ===== =====
Start training: KNNBaseline
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9253
Algorithm KNNBaseline finished with 0.18 mins
===== ===== ===== =====
Start training: KNNBasic
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9617
Algorithm KNNBasic finished with 0.15 mins
===== ===== ===== =====
Start training: KNNWithMeans
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9482
Algorithm KNNWithMeans finished with 0.16 mins
===== ===== ===== =====
Start training: KNNWithZScore
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9482
Algorithm KNNWithZScore finished with 0.17 mins
===== ===== ===== =====
Start training: BaselineOnly
Estimating biases using als...
RMSE: 0.9360
Algorithm BaselineOnly finished with 0.02 mins
===== ===== ===== =====
Start training: CoClustering
RMSE: 0.9564
Algorithm CoClustering finished with 0.07 mins
===== ===== ===== =====
```

In [13]:

```
index = 2
data_name = data_names[index]
data_train = data_train_list[index]
data_test = data_test_list[index]
all_results = {}
print("Start training on dataset: {}".format(data_name))
for algorithm_name in algorithms.keys():
    result = train_single_algorithm(algorithm_name, data_train, data_test)
    all_results[algorithm_name] = result
    print("===== ===== ===== =====")
all_results_all_dataset[data_name] = all_results
```

```
Start training on dataset: Age1
Start training: SVD
RMSE: 1.0627
Algorithm SVD finished with 0.02 mins
===== ===== ===== =====
Start training: SlopeOne
RMSE: 1.0811
Algorithm SlopeOne finished with 0.02 mins
===== ===== ===== =====
Start training: NMF
RMSE: 1.1273
Algorithm NMF finished with 0.02 mins
===== ===== ===== =====
Start training: NormalPredictor
RMSE: 1.6163
Algorithm NormalPredictor finished with 0.00 mins
===== ===== ===== =====
Start training: KNNBaseline
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 1.0557
Algorithm KNNBaseline finished with 0.01 mins
===== ===== ===== =====
Start training: KNNBasic
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 1.1129
Algorithm KNNBasic finished with 0.00 mins
===== ===== ===== =====
Start training: KNNWithMeans
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 1.0804
Algorithm KNNWithMeans finished with 0.00 mins
===== ===== ===== =====
Start training: KNNWithZScore
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 1.0787
Algorithm KNNWithZScore finished with 0.01 mins
===== ===== ===== =====
Start training: BaselineOnly
Estimating biases using als...
RMSE: 1.0605
Algorithm BaselineOnly finished with 0.00 mins
===== ===== ===== =====
Start training: CoClustering
RMSE: 1.1238
Algorithm CoClustering finished with 0.01 mins
===== ===== ===== =====
```

In [14]:

```
index = 3
data_name = data_names[index]
data_train = data_train_list[index]
data_test = data_test_list[index]
all_results = {}
print("Start training on dataset: {}".format(data_name))
for algorithm_name in algorithms.keys():
    result = train_single_algorithm(algorithm_name, data_train, data_test)
    all_results[algorithm_name] = result
    print("===== ===== ===== =====")
all_results_all_dataset[data_name] = all_results
```

```
Start training on dataset: Age18
Start training: SVD
RMSE: 0.9407
Algorithm SVD finished with 0.14 mins
===== ===== ===== =====
Start training: SlopeOne
RMSE: 0.9526
Algorithm SlopeOne finished with 0.14 mins
===== ===== ===== =====
Start training: NMF
RMSE: 0.9666
Algorithm NMF finished with 0.14 mins
===== ===== ===== =====
Start training: NormalPredictor
RMSE: 1.5729
Algorithm NormalPredictor finished with 0.01 mins
===== ===== ===== =====
Start training: KNNBaseline
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9385
Algorithm KNNBaseline finished with 0.12 mins
===== ===== ===== =====
Start training: KNNBasic
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9824
Algorithm KNNBasic finished with 0.10 mins
===== ===== ===== =====
Start training: KNNWithMeans
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9621
Algorithm KNNWithMeans finished with 0.10 mins
===== ===== ===== =====
Start training: KNNWithZScore
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9600
Algorithm KNNWithZScore finished with 0.11 mins
===== ===== ===== =====
Start training: BaselineOnly
Estimating biases using als...
RMSE: 0.9525
Algorithm BaselineOnly finished with 0.01 mins
===== ===== ===== =====
Start training: CoClustering
RMSE: 0.9641
Algorithm CoClustering finished with 0.05 mins
===== ===== ===== =====
```

In [15]:

```
index = 4
data_name = data_names[index]
data_train = data_train_list[index]
data_test = data_test_list[index]
all_results = {}
print("Start training on dataset: {}".format(data_name))
for algorithm_name in algorithms.keys():
    result = train_single_algorithm(algorithm_name, data_train, data_test)
    all_results[algorithm_name] = result
    print("===== ===== ===== =====")
all_results_all_dataset[data_name] = all_results
```

```
Start training on dataset: Age25
Start training: SVD
RMSE: 0.8838
Algorithm SVD finished with 0.30 mins
===== ===== ===== =====
Start training: SlopeOne
RMSE: 0.9048
Algorithm SlopeOne finished with 0.35 mins
===== ===== ===== =====
Start training: NMF
RMSE: 0.9158
Algorithm NMF finished with 0.31 mins
===== ===== ===== =====
Start training: NormalPredictor
RMSE: 1.5201
Algorithm NormalPredictor finished with 0.02 mins
===== ===== ===== =====
Start training: KNNBaseline
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.8909
Algorithm KNNBaseline finished with 0.40 mins
===== ===== ===== =====
Start training: KNNBasic
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9243
Algorithm KNNBasic finished with 0.35 mins
===== ===== ===== =====
Start training: KNNWithMeans
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9207
Algorithm KNNWithMeans finished with 0.37 mins
===== ===== ===== =====
Start training: KNNWithZScore
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9196
Algorithm KNNWithZScore finished with 0.40 mins
===== ===== ===== =====
Start training: BaselineOnly
Estimating biases using als...
RMSE: 0.9064
Algorithm BaselineOnly finished with 0.03 mins
===== ===== ===== =====
Start training: CoClustering
RMSE: 0.9234
Algorithm CoClustering finished with 0.11 mins
===== ===== ===== =====
```

In [16]:

```
index = 5
data_name = data_names[index]
data_train = data_train_list[index]
data_test = data_test_list[index]
all_results = {}
print("Start training on dataset: {}".format(data_name))
for algorithm_name in algorithms.keys():
    result = train_single_algorithm(algorithm_name, data_train, data_test)
    all_results[algorithm_name] = result
    print("===== ===== ===== =====")
all_results_all_dataset[data_name] = all_results
```

```
Start training on dataset: Age35
Start training: SVD
RMSE: 0.8879
Algorithm SVD finished with 0.15 mins
===== ===== ===== =====
Start training: SlopeOne
RMSE: 0.8930
Algorithm SlopeOne finished with 0.17 mins
===== ===== ===== =====
Start training: NMF
RMSE: 0.9087
Algorithm NMF finished with 0.16 mins
===== ===== ===== =====
Start training: NormalPredictor
RMSE: 1.4544
Algorithm NormalPredictor finished with 0.01 mins
===== ===== ===== =====
Start training: KNNBaseline
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.8829
Algorithm KNNBaseline finished with 0.13 mins
===== ===== ===== =====
Start training: KNNBasic
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9186
Algorithm KNNBasic finished with 0.10 mins
===== ===== ===== =====
Start training: KNNWithMeans
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9072
Algorithm KNNWithMeans finished with 0.11 mins
===== ===== ===== =====
Start training: KNNWithZScore
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9070
Algorithm KNNWithZScore finished with 0.12 mins
===== ===== ===== =====
Start training: BaselineOnly
Estimating biases using als...
RMSE: 0.8923
Algorithm BaselineOnly finished with 0.01 mins
===== ===== ===== =====
Start training: CoClustering
RMSE: 0.9134
Algorithm CoClustering finished with 0.06 mins
===== ===== ===== =====
```

In [17]:

```
index = 6
data_name = data_names[index]
data_train = data_train_list[index]
data_test = data_test_list[index]
all_results = {}
print("Start training on dataset: {}".format(data_name))
for algorithm_name in algorithms.keys():
    result = train_single_algorithm(algorithm_name, data_train, data_test)
    all_results[algorithm_name] = result
    print("===== ===== ===== =====")
all_results_all_dataset[data_name] = all_results
```

```
Start training on dataset: Age45
Start training: SVD
RMSE: 0.8953
Algorithm SVD finished with 0.06 mins
===== ===== ===== =====
Start training: SlopeOne
RMSE: 0.9044
Algorithm SlopeOne finished with 0.07 mins
===== ===== ===== =====
Start training: NMF
RMSE: 0.9232
Algorithm NMF finished with 0.07 mins
===== ===== ===== =====
Start training: NormalPredictor
RMSE: 1.4343
Algorithm NormalPredictor finished with 0.00 mins
===== ===== ===== =====
Start training: KNNBaseline
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.8883
Algorithm KNNBaseline finished with 0.03 mins
===== ===== ===== =====
Start training: KNNBasic
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9390
Algorithm KNNBasic finished with 0.02 mins
===== ===== ===== =====
Start training: KNNWithMeans
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9068
Algorithm KNNWithMeans finished with 0.03 mins
===== ===== ===== =====
Start training: KNNWithZScore
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9060
Algorithm KNNWithZScore finished with 0.03 mins
===== ===== ===== =====
Start training: BaselineOnly
Estimating biases using als...
RMSE: 0.8928
Algorithm BaselineOnly finished with 0.00 mins
===== ===== ===== =====
Start training: CoClustering
RMSE: 0.9319
Algorithm CoClustering finished with 0.03 mins
===== ===== ===== =====
```

In [18]:

```
index = 7
data_name = data_names[index]
data_train = data_train_list[index]
data_test = data_test_list[index]
all_results = {}
print("Start training on dataset: {}".format(data_name))
for algorithm_name in algorithms.keys():
    result = train_single_algorithm(algorithm_name, data_train, data_test)
    all_results[algorithm_name] = result
    print("===== ===== ===== =====")
all_results_all_dataset[data_name] = all_results
```

```
Start training on dataset: Age50
Start training: SVD
RMSE: 0.9004
Algorithm SVD finished with 0.05 mins
===== ===== ===== =====
Start training: SlopeOne
RMSE: 0.9080
Algorithm SlopeOne finished with 0.06 mins
===== ===== ===== =====
Start training: NMF
RMSE: 0.9346
Algorithm NMF finished with 0.06 mins
===== ===== ===== =====
Start training: NormalPredictor
RMSE: 1.4272
Algorithm NormalPredictor finished with 0.01 mins
===== ===== ===== =====
Start training: KNNBaseline
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.8917
Algorithm KNNBaseline finished with 0.02 mins
===== ===== ===== =====
Start training: KNNBasic
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9506
Algorithm KNNBasic finished with 0.02 mins
===== ===== ===== =====
Start training: KNNWithMeans
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9103
Algorithm KNNWithMeans finished with 0.02 mins
===== ===== ===== =====
Start training: KNNWithZScore
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9102
Algorithm KNNWithZScore finished with 0.02 mins
===== ===== ===== =====
Start training: BaselineOnly
Estimating biases using als...
RMSE: 0.8957
Algorithm BaselineOnly finished with 0.01 mins
===== ===== ===== =====
Start training: CoClustering
RMSE: 0.9279
Algorithm CoClustering finished with 0.02 mins
===== ===== ===== =====
```

In [19]:

```
index = 8
data_name = data_names[index]
data_train = data_train_list[index]
data_test = data_test_list[index]
all_results = {}
print("Start training on dataset: {}".format(data_name))
for algorithm_name in algorithms.keys():
    result = train_single_algorithm(algorithm_name, data_train, data_test)
    all_results[algorithm_name] = result
    print("===== ===== ===== =====")
all_results_all_dataset[data_name] = all_results
```

```
Start training on dataset: Age56
Start training: SVD
RMSE: 0.8957
Algorithm SVD finished with 0.03 mins
===== ===== ===== =====
Start training: SlopeOne
RMSE: 0.9264
Algorithm SlopeOne finished with 0.02 mins
===== ===== ===== =====
Start training: NMF
RMSE: 0.9539
Algorithm NMF finished with 0.03 mins
===== ===== ===== =====
Start training: NormalPredictor
RMSE: 1.4106
Algorithm NormalPredictor finished with 0.00 mins
===== ===== ===== =====
Start training: KNNBaseline
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.8913
Algorithm KNNBaseline finished with 0.01 mins
===== ===== ===== =====
Start training: KNNBasic
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9671
Algorithm KNNBasic finished with 0.01 mins
===== ===== ===== =====
Start training: KNNWithMeans
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9150
Algorithm KNNWithMeans finished with 0.01 mins
===== ===== ===== =====
Start training: KNNWithZScore
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9141
Algorithm KNNWithZScore finished with 0.01 mins
===== ===== ===== =====
Start training: BaselineOnly
Estimating biases using als...
RMSE: 0.8850
Algorithm BaselineOnly finished with 0.00 mins
===== ===== ===== =====
Start training: CoClustering
RMSE: 0.9418
Algorithm CoClustering finished with 0.01 mins
===== ===== ===== =====
```

Aggregate the datasets

- female - male
- different ages

In [23]:

```
def combine_result(all_results_list):
    """
    Aggregate the result of different datasets for each algorithm
    """
    algorithms = {'SVD':SVD(), 'SlopeOne':SlopeOne(), 'NMF':NMF(), 'NormalPredictor':NormalPredictor(),
              'KNNBaseline':KNNBaseline(), 'KNNBasic':KNNBasic(), 'KNNWithMeans':KNNWithMeans(),
              'KNNWithZScore':KNNWithZScore(), 'BaselineOnly':BaselineOnly(), 'CoClustering':CoClustering()}
    for algorithm_name in algorithms.keys():
        part1_rmse = 0
        part1_recall = 0
        part2 = 0
        for all_results in all_results_list:
            n_samples = all_results[algorithm_name]['n_samples']
            rmse = all_results[algorithm_name]['rmse']
            recall = all_results[algorithm_name]['recall']
            part1_rmse += rmse ** 2 * n_samples
            part1_recall += recall * n_samples
            part2 += n_samples
        combined_rmse = np.sqrt(part1_rmse / part2)
        combined_recall = part1_recall / part2
        print("{:<15}|Rmse {:.4f}|Recall {:.4f}".format(algorithm_name, combined_rmse, combined_recall))
```

In [24]:

```
# gender combined
results_list = [all_results_all_dataset['Male'], all_results_all_dataset['Female']]
combine_result(results_list)
```

```
SVD            |Rmse 0.8829|Recall 0.5503
SlopeOne       |Rmse 0.9057|Recall 0.5385
NMF            |Rmse 0.9169|Recall 0.5262
NormalPredictor|Rmse 1.5043|Recall 0.3915
KNNBaseline    |Rmse 0.8939|Recall 0.5544
KNNBasic       |Rmse 0.9250|Recall 0.5742
KNNWithMeans   |Rmse 0.9257|Recall 0.5221
KNNWithZScore  |Rmse 0.9259|Recall 0.5256
BaselineOnly   |Rmse 0.9075|Recall 0.5523
CoClustering   |Rmse 0.9190|Recall 0.5448
```

In [25]:

```
# Age combined
results_list = [all_results_all_dataset['Age1'], all_results_all_dataset['Age18'], all_results_all_dataset['Age25'],
               all_results_all_dataset['Age35'], all_results_all_dataset['Age45'], all_results_all_dataset['Age50'],
               all_results_all_dataset['Age56']]
combine_result(results_list)
```

```
SVD            |Rmse 0.9031|Recall 0.5420
SlopeOne       |Rmse 0.9176|Recall 0.5346
NMF            |Rmse 0.9336|Recall 0.5224
NormalPredictor|Rmse 1.5023|Recall 0.3900
KNNBaseline    |Rmse 0.9029|Recall 0.5537
KNNBasic       |Rmse 0.9444|Recall 0.5675
KNNWithMeans   |Rmse 0.9283|Recall 0.5285
KNNWithZScore  |Rmse 0.9273|Recall 0.5326
BaselineOnly   |Rmse 0.9141|Recall 0.5490
CoClustering   |Rmse 0.9367|Recall 0.5407
```

In [ ]: