Построить collaborative filtering рекомендательную систему по фильмам movielens

### 

[Задача](#_a94xmr2cf7nr)

[Обработка данных на вход](#_xcxkmvreh1sw)

[Обработка данных на выход](#_vtp9bfl2gg8a)

[Проверка](#_m4mc0relc5tx)

[Дедлайны](#_838wd1wj4hua)

### 

### Задача

По имеющимся данным о рейтингах фильмов (movielens: 100 000 рейтингов) построить коллаборативную рекомендательную систему.

### Обработка данных на вход

Имеются следующие входные данные:

* таблица users\*films с рейтингами (архив с данными нужно скачать с [grouplens.org](http://grouplens.org/datasets/movielens/) – [zip](http://files.grouplens.org/datasets/movielens/ml-100k.zip), также доступен на кластере в hdfs по пути /recsys/ml-100k). Файл u.data содержит все оценки, а файл u.item список всех фильмов.
* id выданного вам пользователя ([в личном кабинете](http://lk.newprolab.com/lab/lab09))

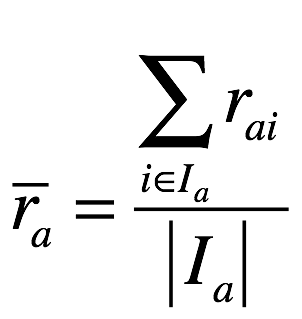
### Обработка данных на выход

Часть1. Найдем основные характеристики данных movielens:

1. Найдите количество всех пользователей и количество всех фильмов в данных (общее число пользователей и фильмов в датасете).
2. Сколько пользователь в среднем ставит рейтингов? (количество рейтингов/количество пользователей) **– average\_user\_ratings**
3. Сколько фильм в среднем имеет рейтингов (количество рейтингов/количество фильмов)? **– average\_film\_ratings**
4. Найдите процент заполненных ячеек в данных (количество рейтингов/(количество пользователей\*количество фильмов)) **– completeness**
5. Какой тип CF вы бы использовали на данном датасете? Ответ не оценивается:
   1. user-user
   2. item-item

Часть 2. User-user CF:

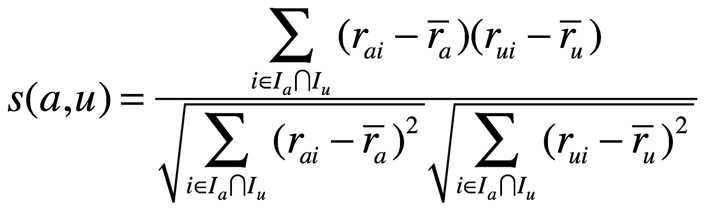
1. Для каждого пользователя найдите его средний рейтинг (сумма рейтингов пользователя/количество рейтингов пользователя). Здесь - множество фильмов, по которым у пользователя есть рейтинги . Здесь и далее обозначает количество элементов в множестве .



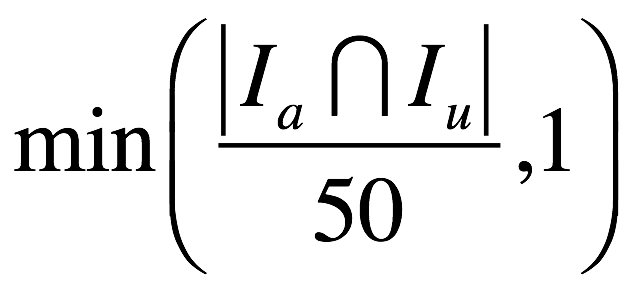
Для выданного вам пользователя (ID в личном кабинете):

1. Посчитайте меру близости Пирсона выданного вам пользователя со всеми остальными пользователями. Обратите внимание, что корреляция Пирсона считается только на пересечении, то есть вклад вносят только фильмы, оцененные совместно (- множества оцененных пользователями и фильмов). Корреляция с константой (ситуация, когда у пользователя все оценки одинаковые) равна нулю.

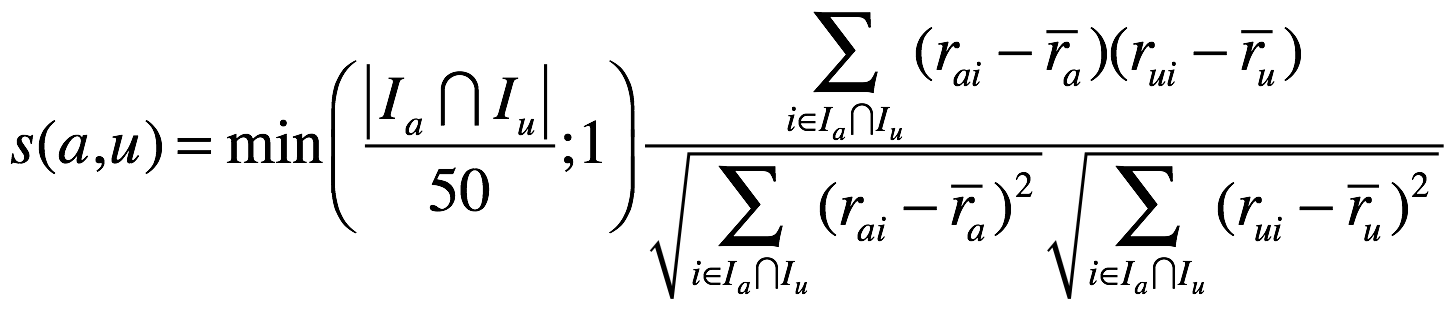
Формат: ID пользователя; корреляция Пирсона.



1. Посчитайте поправочный коэффициент для корреляции Пирсона на нехватку данных

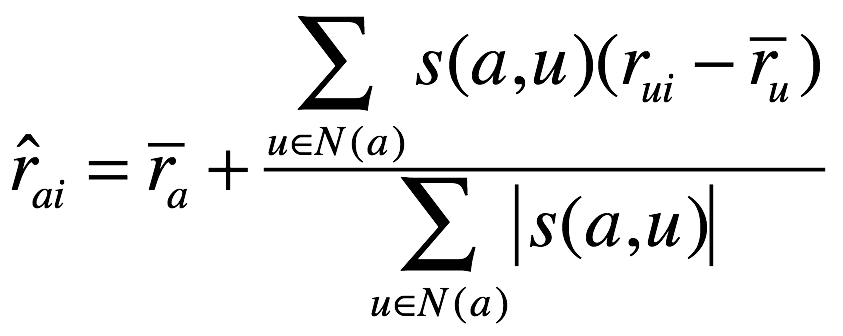


1. Найдите 30 ближайших пользователей-соседей данного пользователя (**pearson\_neighbours**), используя поправленную корреляцию Пирсона:



Дальше мы будем использовать только этих соседей для прогноза. Заметим, что строя предсказание для некоторого фильма, мы из фиксированных 30 ближайших пользователей-соседей выбираем только тех, у кого есть оценки для данного фильма, то есть используемых пользователей-соседей для прогноза может быть сильно меньше. Можно находить k ближайших пользователей-соседей, у которых есть оценки по интересующему нас фильму, но это вычислительно гораздо сложнее, поэтому целесообразность этого следует проверять на практике (в рамках данной лабораторной, это лишнее усложнение).

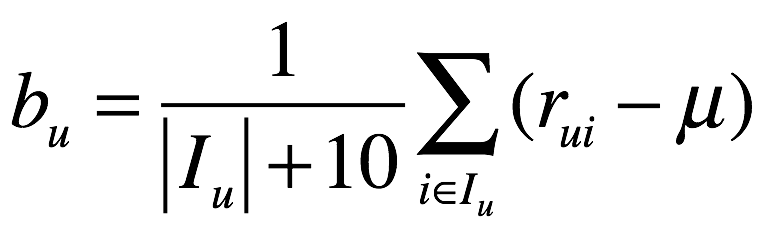
1. Для всех фильмов найдите прогноз оценки по формуле ниже. Здесь - множество пользователей-соседей, - мера близости пользователей из предыдущих пунктов, - модуль меры близости.



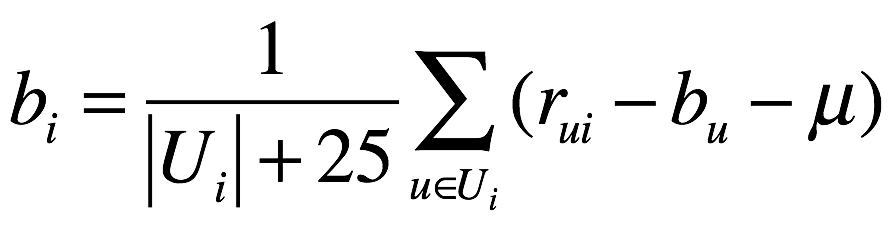
1. Рекомендуйте пользователю 10 фильмов (**pearson\_top10**), которые он еще не оценивал, с самыми высокими прогнозами из предыдущего пункта.
2. Сравните результат с топ-10 фильмами, полученными по нижней границе доверительного интервала Wilson - поле “top10\_lower” из Лабораторной работы №7. Найдите количество общих фильмов в этих двух десятках.

Часть 3. Базовые предикторы:

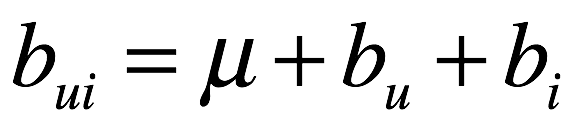
1. Глобальное среднее (**average\_rating**) по всему датасету. Сумма всех оценок по всем фильмам/Количество всех оценок по всем фильмам. (Лабораторная №9 пункт 5 первой части)
2. Базовый предиктор для каждого пользователя (суммирование по фильмам, оцененным данным пользователем). Здесь - множество фильмов, по которым у пользователя есть рейтинги, а их количество.



1. Базовый предиктор для каждого фильма (суммирование по пользователям, поставившим оценку данному фильму). Здесь - множество пользователей, которые оценили данный фильм, а их количество.

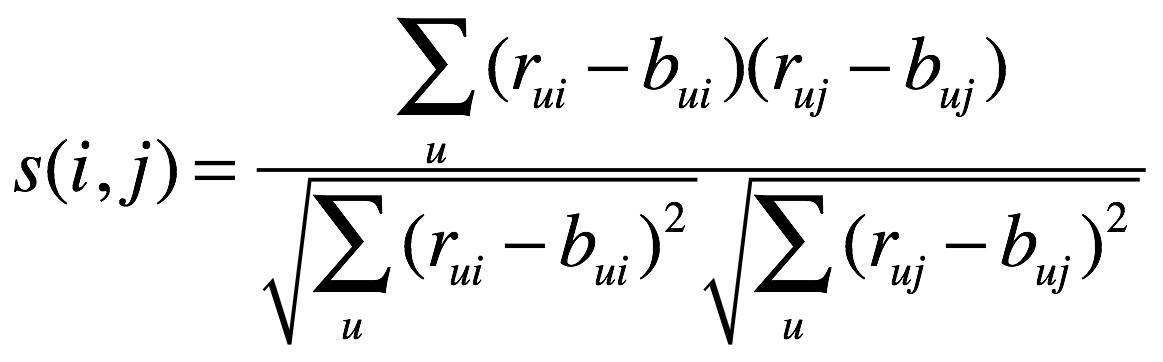


1. Базовый предиктор для каждого пользователя и каждого фильма

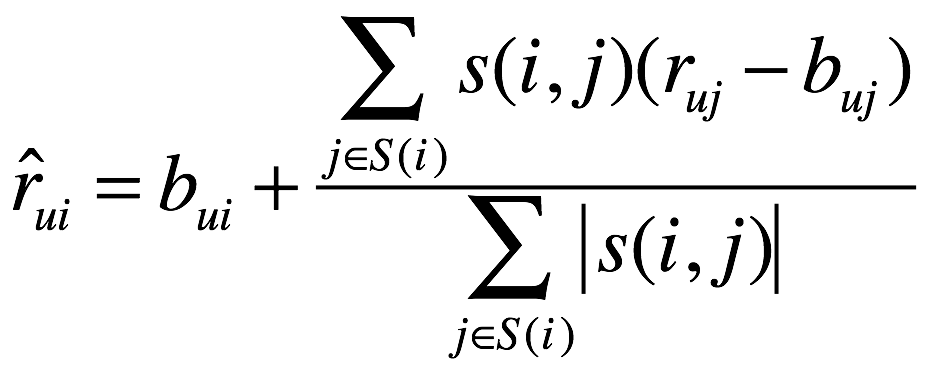


Часть 4. Item-item CF:

1. Вычесть из всех рейтингов базовый предиктор из пункта 4, часть 3 (для всей таблицы рейтингов). Если рейтинга нет, то можно поставить 0.
2. Найдите попарные меры близости (косинус) для всех фильмов, используя очищенные оценки из пункта 1, часть 4. Суммирование идет по всем пользователям.



1. Для каждого фильма, по которому у данного пользователя не стоит рейтинг, найдите
   1. 30 ближайших фильмов-соседей для этого фильма (среди всех фильмов, а не фильмов, оцененных пользователем)
   2. прогноз оценки пользователя по формуле (базовый предиктор из пункта 4, часть 3). Здесь - множество фильмов-соседей для фильма , по которым у данного пользователя есть оценка.



Заметим, что суммирование идет только по тем фильмам-соседям, которые оценил пользователь.

1. Рекомендуйте пользователю 10 фильмов (**predicators\_top10**) с самыми высокими оценками из пункта 3, часть 4.
2. При подсчете прогноза по формуле из пункта 3, часть 4 выкиньте всех соседей с отрицательной близостью.
3. Рекомендуйте пользователю 10 фильмов (**predicators\_positive\_top10**) с самыми высокими оценками из пункта 5, часть 4.

Посмотрите совпадают ли топы в пунктах 4 и 6 (для себя).

1. Найдите количество общих фильмов в топ-10 из пункта 6, часть 2 и топ-10 из пункта 6, часть 4.

**Проверка**

Результат следует сохранить в файл **lab09.json** в своей домашней директории в следующем формате:

|  |
| --- |
| {  "average\_film\_ratings": 89.2222,  "average\_rating": 3.1111,  "average\_user\_ratings": 156.4444,  "completeness": 0.093434,  "pearson\_neighbours": [123, 768, 174, 82, 457, 600, 83, 577, 22, 26, 348, 397, 615, 222, 881, 57, 719, 847, 831, 541, 452, 69, 87, 619, 456, 95, 789, 402, 303, 622],  "pearson\_top10": [1137, 1167, 481, 525, 165, 494, 863, 1427, 968, 1282],  "predicators\_positive\_top10": [489, 604, 1449, 611, 656, 170, 847, 793, 615, 963],  "predicators\_top10": [123, 604, 124, 456, 656, 170, 847, 793, 615, 963]  } |

При равенстве соответствующих метрик сортировать необходимо по возрастанию id, рассматривать как число.

Округление метрик - до 4 знаков после запятой.

Для получения True по четырем последним ключам точность совпадения должна быть 0.8.

Проверка происходит через личный кабинет.

### Дедлайны

Дедлайн: ср, 7 декабря 23:59