Построение рекомендательной системы фильмов

Рекомендательные системы – это популярная и очень полезная возможность на многих сайтах, предлагающих музыку, книги, фильмы, интересные ссылки, электронику и прочие товары. Некоторые рекомендательные системы позволяют даже подбирать вторую половинку на сайтах знакомств. Реализация рекомендательных систем относительно проста, но в то же время даёт значительные конкурентные преимущества.

Среди известных сервисов, содержащих рекомендательные системы, можно привести в пример:

- Кинопоиск с системой рекомендации музыки;
- Amazon, рекомендующий книги и фильмы;
- AliExpress с системой рекомендации товаров;
- Last.fm с системой рекомендации музыки;
- Coursera – рекомендации бесплатных курсов.

Сайты собирают информацию по-разному. Где-то учитываются просмотры и покупки товаров, какие-то сервисы учитывают оценки товаров пользователями по пятибалльной шкале.

- Билеты на концерт: купил (1), не купил (0)
- Онлайн-покупки: купил (2), просмотрел (1), не купил (0)
- Новостная статья: понравилась (1), не читал (0), не понравилась (-1)

Рекомендательные системы строятся на одной простой идее. Среди множества пользователей выбираются те, чьи предпочтения наиболее соответствуют Вашим. Затем Вам будут предложены те новые для Вас товары и услуги, которые были высоко оценены этими пользователями.

Для начала нам потребуется создать словарь критиков и их оценки для некоторых популярных фильмов. Данные представлены в таблице 1.

Таблица 1 – Данные для словаря критиков

	Зима в Простоквашино	Каникулы в Простоквашино	Ёжик в тумане	Винни-Пух	Ну, погоди!	Котёнок по имени Гав
Кот Матроскин	2,5	3,5	3,0	3,5	2,5	3,0
Пёс Шарик	3,0	3,5	1,5	5,0	3,5	3,0
Почтальон Печкин	2,5	3,0	–	3,5	–	4,0
Корова Мурка	–	3,5	3,0	4,0	2,5	4,5
Телёнок Гаврюша	3,0	4,0	2,0	3,0	2,0	3,0
Галчонок	3,0	4,0	–	5,0	3,5	3,0
Дядя Фёдор	–	4,5	–	4,0	1,0	–

Создайте на жёстком диске файл **recommendations.py** со следующим содержимым:

```
# словарь критиков и их оценок для небольшого числа мультфильмов
critics = {
    'Кот Матроскин': {
        'Зима в Простоквашино': 2.5,
        'Каникулы в Простоквашино': 3.5,
        'Ёжик в тумане': 3.0,
        'Винни-Пух': 3.5,
        'Ну, погоди!': 2.5,
        'Котёнок по имени Гав': 3.0
    },
    'Пёс Шарик': {
        'Зима в Простоквашино': 3.0,
        'Каникулы в Простоквашино': 3.5,
        'Ёжик в тумане': 1.5,
        'Винни-Пух': 5.0,
        'Котёнок по имени Гав': 3.0,
        'Ну, погоди!': 3.5
    },
    'Почтальон Печкин': {
        'Зима в Простоквашино': 2.5,
        'Каникулы в Простоквашино': 3.0,
        'Винни-Пух': 3.5,
        'Котёнок по имени Гав': 4.0
    },
    'Корова Мурка': {
        'Каникулы в Простоквашино': 3.5,
        'Ёжик в тумане': 3.0,
        'Котёнок по имени Гав': 4.5,
        'Винни-Пух': 4.0,
        'Ну, погоди!': 2.5
    },
    'Телёнок Гаврюша': {
        'Зима в Простоквашино': 3.0,
        'Каникулы в Простоквашино': 4.0,
        'Ёжик в тумане': 2.0,
        'Винни-Пух': 3.0,
        'Котёнок по имени Гав': 3.0,
        'Ну, погоди!': 2.0
    },
    'Галчонок': {
        'Зима в Простоквашино': 3.0,
        'Каникулы в Простоквашино': 4.0,
        'Котёнок по имени Гав': 3.0,
        'Винни-Пух': 5.0,
        'Ну, погоди!': 3.5
    },
    'Дядя Фёдор': {
        'Каникулы в Простоквашино': 4.5,
        'Ну, погоди!': 1.0,
        'Винни-Пух': 4.0
    }
}
```

Запустите интерпретатор **Python** через меню «**Пуск**» (IDLE или command line), перейдите в каталог, в котором расположен файл **recommendations.py** с помощью команд:

```
>>> import os
>>> os.chdir("<путь>")
```

Не забывайте экранировать обратные слеши.

Проверить текущий рабочий каталог можно с помощью команды:

```
>>> os.getcwd()
```

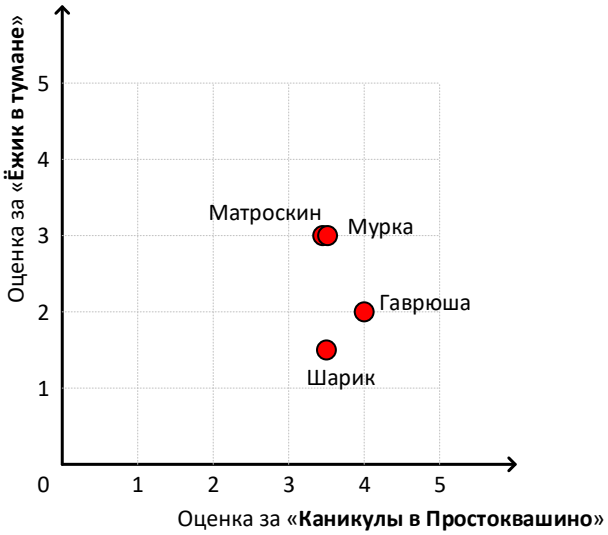
В интерпретаторе **Python** выполните следующий код:

```
>>> from recommendations import critics
>>> critics['Дядя Фёдор']['Ну, погоди!']
1.0
>>> critics['Кот Матроскин']
{'Котёнок по имени Гав': 3.0, 'Ну, погоди!': 2.5, 'Ёжик в тумане': 3.0, 'Винни-Пух': 3.5,
'Зима в Простоквашино': 2.5, 'Каникулы в Простоквашино': 3.5}
```

Поиск пользователей со схожими предпочтениями

Теперь необходимо найти критиков со сходными предпочтениями. Для этого есть разные способы. Мы рассмотрим оценку по евклидову расстоянию и с помощью коэффициента корреляции Пирсона.

Для начала давайте изобразим на плоскости оценки всех критиков по фильмам «Каникулы в Простоквашино» и «Ёжик в тумане». Мы будем отмечать только тех критиков, которые поставили оценки обоим этим бестселлерам.



На диаграмме приведены только два показателя, так как мы отображаем данные на двумерной плоскости, но подход работает для произвольного количества показателей.

⇒ **Выполнить Задание 1**

Меру близости двух критиков можно вычислить как **евклидово расстояние**:

$$P = \sqrt{\sum_{i=1}^{N}(c_{1i} - c_{2i})^2} \qquad (1)$$

где c_{1i} – оценка первым критиком i-го фильма, c_{2i} – оценка вторым критиком i-го фильма, N – количество фильмов, которые были оценены обоими критиками.

Видно, что чем больше совпадают вкусы критиков, тем меньше расстояние между ними. Но хотелось бы получить меру, которая тем выше, чем ближе предпочтения критиков. Для этого можно использовать следующую формулу:

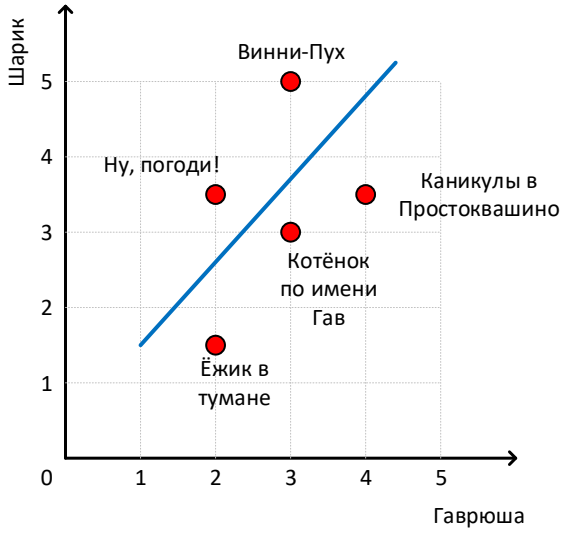
$$P = \frac{1}{1 + \sqrt{\sum_{i=1}^{N}(c_{1i} - c_{2i})^2}} \qquad (2)$$

При этом при совпадении оценок двух критиков метрика даст 1, а при несовпадении – оценку, близкую к нулю.

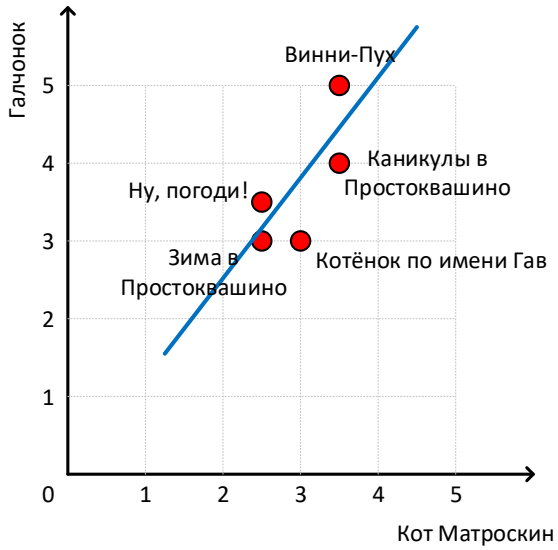
⇒ **Выполнить Задание 2**

Второй способ оценки схожести интересов двух пользователей – использование **коэффициента корреляции Пирсона**. Этот способ более сложный, чем предыдущий, но он хорошо работает в том случае, когда пользователь ставит систематически заниженные или завышенные оценки. Коэффициент корреляции – это мера того, насколько хорошо два набора данных ложатся на прямую.

Построим на графике для двух критиков – Шарика и Гаврюши – оценки общих фильмов, а также прямую линию, которая называется линией наилучшего приближения, так как проходит максимально близко ко всем точкам на графике. Если бы оба критика выставили одинаковые оценки всем фильмам, линия была бы диагональю, проходящей через все точки. Это дало бы коэффициент корреляции 1.



Для приведённого рисунка коэффициент корреляции Пирсона для Шарика и Гаврюши равен 0.41. Рассмотрим теперь двух других критиков – Кота Матроскина и Галчонка (см. рисунок ниже).



Коэффициент корреляции для этих критиков равен 0.75, что означает, что их вкусы в значительной степени совпадают.

$$P = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{\left(\sum X^2 - \frac{(\sum X)^2}{N}\right) \cdot \left(\sum Y^2 - \frac{(\sum Y)^2}{N}\right)}} \quad (3)$$

⇒ **Выполнить Задание 3**

Ранжирование критиков

После того, как у нас появилась возможность оценить схожесть вкусов двух критиков, необходимо определить критика с наиболее похожим вкусом по отношению к заданному критику. Для этого необходимо сравнить заданного критика со всеми остальными, измерить метрику близости их предпочтений и вывести того критика, чей вкус наиболее совпадает со вкусом заданного критика.

⇒ **Выполнить Задание 4**

Задание на практику

Задание 1. Отметить точки критиков на координатной плоскости двух фильмов. Названия фильмов следует брать из таблицы:

	Зима в Простокваши но	Каникулы в Простокваши но	Ёжик в тумане	Винни-Пух	Ну, погоди!	Котёнок по имени Гав
Зима в Простоквашино		1	2	3	4	5
Каникулы в Простоквашино	1		6	7	8	9
Ёжик в тумане	2	6		10	11	12
Винни-Пух	3	7	10		13	14
Ну, погоди!	4	8	11	13		15
Котёнок по имени Гав	5	9	12	14	15	

В ячейках таблицы указан номер варианта.

Задание 2. Написать функцию **`sim_distance`**, вычисляющую метрику схожести по формуле 2. Функция получает 3 параметра: словарь **`critics`**, имя первого критика, имя второго критика. Если не существует ни одного фильма, который был оценён обоими критиками сразу, функция должна вернуть 0. Для всех фильмов, которые были оценены обоими критиками, функция должна возвращать метрику схожести, рассчитанную по формуле 2. Привести 2 примера ручного расчёта метрики, результаты вызова функции для этих примеров и код функции. Функция должна располагаться в файле **`recommendations.py`**.

Для успешного выполнения задания необходимо познакомиться с основами языка **Python**, в частности с написанием и вызовом функций, импортом кода из внешних источников, конструкциями циклов и условий, функциями определения длины массива, суммы элементов массива, возведения в степень и извлечения квадратного корня. Для экспериментов с конструкциями языка удобнее всего использовать консоль **Python**.

Задание 3. Написать функцию **`sim_pearson`**, вычисляющую метрику схожести, основанную на коэффициенте корреляции Пирсона, по формуле 3. Функция получает те же 3 параметра, что и функция **`sim_distance`**. Следует обратить внимание, что расчёт должен производиться только по тем фильмам, которые были оценены одновременно обоими критиками. Если таких фильмов не оказалось, функция должна вернуть 0. Если знаменатель в формуле 3 равен нулю, функция так же должна вернуть 0. Функция так же должна располагаться в файле **`recommendations.py`**. Необходимо привести 2 примера вызова функции и её код.

Задание 4. Написать функцию **`top_matches`**, которая принимает 2 параметра: словарь критиков и имя критика, для которого ищем похожих. Функция должна вычислить метрику схожести (евклидово расстояние или коэффициент корреляции Пирсона) между заданным критиком и всеми остальными критиками, отсортировать их по убыванию и вернуть отсортированный список. Список должен содержать имена критиков и оценки схожести. Обратите внимание, что сравнивать критика самого с собой не нужно. Функция **`top_matches`** должна располагаться в файле **`recommendations.py`**. Необходимо привести вызов функции для критика, номер которого

в словаре определяется как остаток от деления номера студента в списке на 7. Найти наиболее похожего и наименее похожего пользователей. Изобразить на двух рисунках оценки фильмов заданного критика и наиболее похожего и заданного критика и наименее похожего. Нарисовать линии наилучшего приближения (на глаз).