# Лабораторная работа №1:

# Выработка рекомендаций

## Цель:

Целью данной работы является получение базовых навыков работы с массивами статистических данных средствами языка Python. А так же, изучение и апробирование на практике методик выработки рекомендаций в рамках системы с определёнными параметрами.

**Справка:** интерпретатор для Python можно получить по адресу https://www.python.org/downloads/
В качестве интегрированной системы разработки могут быть использованы:
- PyCharm Community Edition: http://www.jetbrains.com/pycharm/download/
- Python Tools for Visual Studio: http://pytools.codeplex.com/
Либо редактор текста с подсветкой синтаксиса Notepad++: http://notepad-plus-plus.org/

## Часть 1: получение данных

Для выполнения лабораторной работы необходимы тестовые наборы данных, которые могут быть получены по следующим ссылкам:

Данные об оценках фильмов: https://grouplens.org/datasets/movielens/ (MovieLens 100k)

Данные о прослушивании музыкальных композиций: https://grouplens.org/datasets/hetrec-2011/

**Структура данных в архиве ml-100k**:
u.item – содержит информацию о фильмах. Для выполнения лабораторной работы достаточно получить первые два поля в каждой строке представленных в формате идентификационный `номер фильма|название` (разделителем является символ “|”)
u.data – содержит информацию об оценках фильмов. Данные представлены в формате `номер пользователя|номер фильма|оценка|время` (разделителем является tab, символ “\t”)
Более подробную информацию о содержимом архива можно получить в файле Readme

Для того, что бы получить данные о номере фильма и его названии, можно использовать следующий программный код:

```
movies = {}
for line in open(path+'\u.item'):
    (movie_id, movie_title) = line.split('|')[0:2]
    movies[movie_id] = movie_title
```

Таким образом, будет создан словарь `movies`, ключом которого будет номер фильма, а значением его название.

Переменная `path` должна содержать путь к директории, в которой расположен файл

Функция `line.split` описывает разделитель и число элементов строки, которое необходимо считать.

Получение информации о рейтингах фильмов будет выглядеть следующим образом:

```
ratings = {}
for line in open(path+'\u.data'):
    (user_id, movie_id, rating) = line.split('\t')[0:3]
    ratings.setdefault(user_id, {})
    ratings[user_id][movies[movie_id]] = float(rating)
```

При этом `ratings` будет вложенным словарём, в котором первым ключом будет являться номер пользователя, вторым название фильма, а значением оценка.

**Задание:** необходимо реализовать функцию получения заданного числа наборов данных из файлов с данными о названии фильмов и оценках пользователей удовлетворяющую следующим условиям:

- число необходимых наборов должно передаваться в качестве параметра функции
- по умолчанию, функция должна считывать всё содержимое полностью
- в считанном наборе данных должны быть записи, имеющие общие ключи “название фильма”

## Часть 2: Оценка подобия

В данной лабораторной, под подобием, понимается степень корреляции пересекающегося подмножества оценок фильмов для разных пользователей.

**Евклидова метрика:**

Методика определения расстояния между парой точек в N-мерном пространстве основанная на теореме Пифагора о соотношении сторон прямоугольного треугольников.

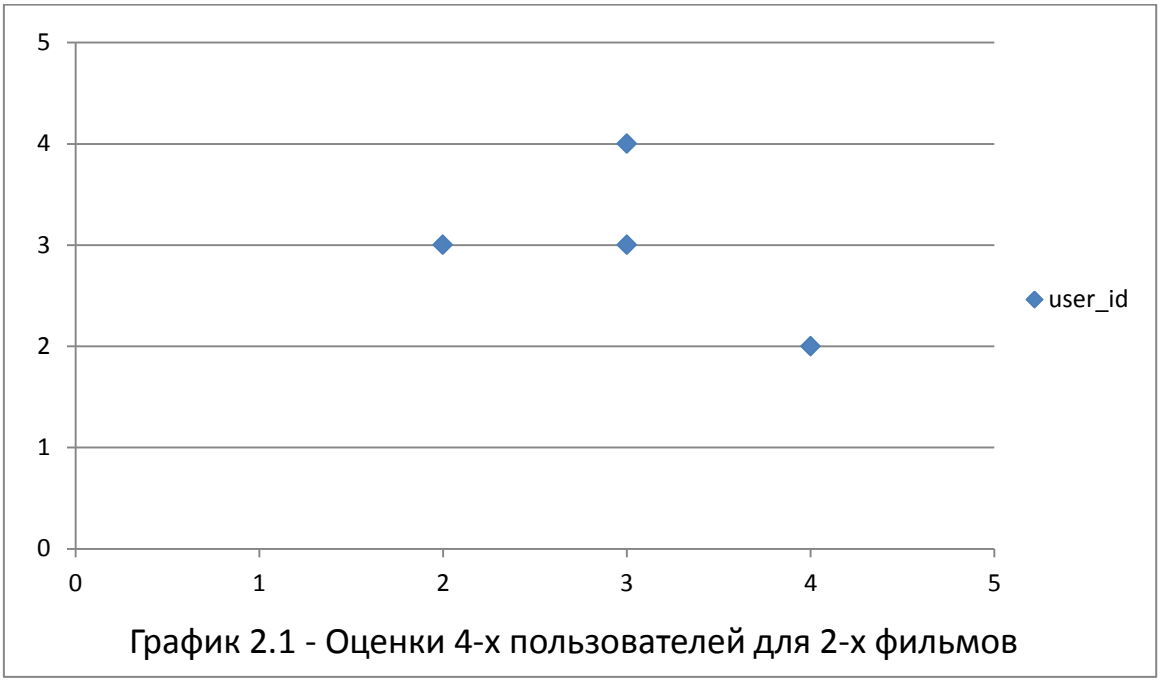


График 2.1 - Оценки 4-х пользователей для 2-х фильмов

На графике выше, показано визуальное представление распределения оценок четырёх пользователей для двух фильмов. Согласно евклидовой метрике, для того что бы определить расстояние между двумя точками, необходимо посчитать корень от суммы квадратов разностей проекций оценок на оси.

В общем виде, формула выглядит следующим образом:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 \dots (p_n - q_n)^2} = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

Формула 2.1 – евклидово расстояние

Где p и q – оценки, отложенные по одной оси.

Полученная величина будет тем меньше, чем ближе расположены оценки пользователей на всей системе координат.

Для того, что бы на основании расстояния между пользователями вычислить коэффициент подобия, можно воспользоваться следующим выражением.

$$k = \frac{1}{(1 + \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2})}$$

Формула 2.2 – коэффициент подобия по евклидовой метрике

Полученное значение всегда будет в диапазоне [0, 1], где 1 – полное совпадение оценок пользователей.

Реализация евклидовой метрики может выглядеть следующим образом:

```
s = 0
for item in rate[pers1]:
    if item in rate[pers2]:
        s += 1

if s < 1:
    return 0

sq_sum = 0.0

for item in rate[pers1]:
    if item in rate[pers2]:
        sq_sum += pow(rate[pers1][item] - rate[pers2][item], 2)

if sqrt(sq_sum) < 1:
    sq_sum = 1
else:
    sq_sum = 1/sqrt(sq_sum)

return sq_sum
```

Вместо прибавления единицы для устранения ошибки деления на ноль, использована проверка значения.

**Расстояние городских кварталов:**

Метрика, подобная евклидовой, однако рассчитывается для пространства n-мерных векторов. В общем виде, описывается формулой:

$$d(p,q) = \|p - q\| = \sum_{i=1}^{n} |p_i - q_i|$$

Формула 2.3 – расстояние городских кварталов

Где $p = (p_1, p_2, \dots, p_{n)}$ и $q = (q_1, q_2, \dots, q_n)$ – вектора.

**Коэффициент корреляции Пирсона:**

В отличие от эвклидовой метрики и расстояния городских кварталов, коэффициент Пирсона характеризует существование линейной зависимости между двумя величинами.

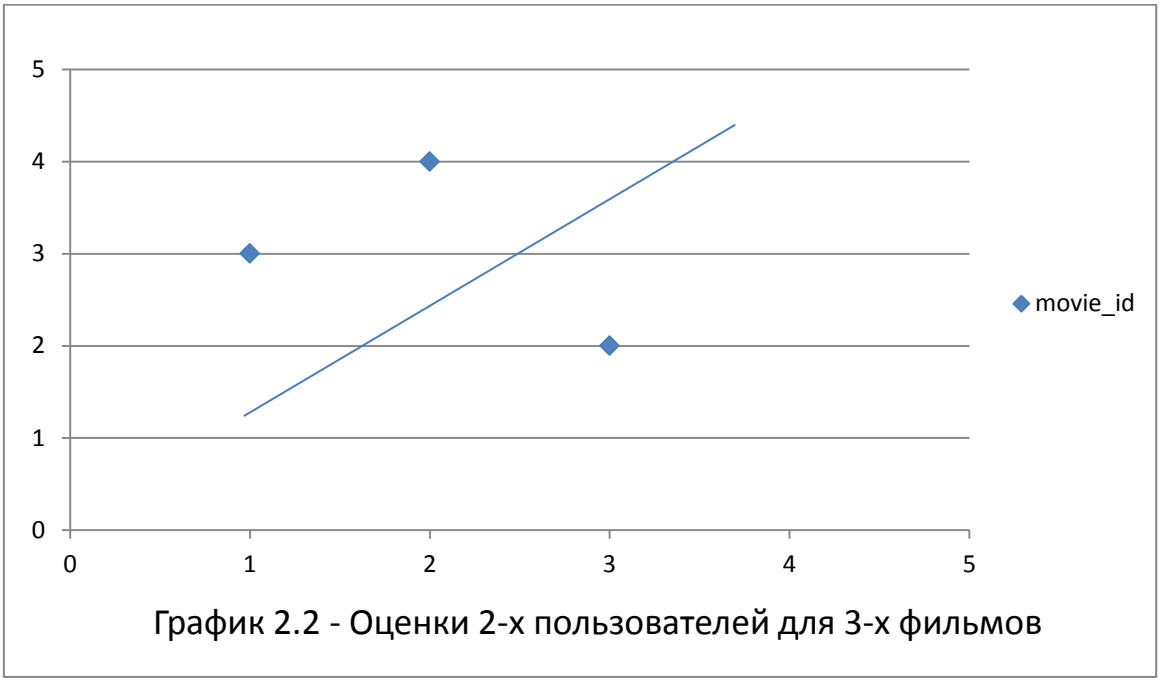


График 2.2 - Оценки 2-х пользователей для 3-х фильмов

На представленном выше графике отмечены оценки двух пользователей для трёх фильмов, а так же линия наилучшего приближения. Линией наилучшего приближения, называют линию, которая проходит максимально близко к имеющимся точкам.

Степень совпадения значений отмеченных на графике и линии, определяется при помощи коэффициента Пирсона.

В общем виде, формула для расчёта коэффициента записывается как:

$$r = \frac{\sum(x_i * y_i) - \sum x_i * \frac{\sum y_i}{n}}{\sqrt{\left(\sum x_i^2 - \frac{(\sum x_i)^2}{n}\right) * \left(\sum y_i^2 - \frac{(\sum y_i)^2}{n}\right)}}$$

Формула 2.4 – коэффициент Пирсона

Коэффициент r принимает значения в диапазоне [-1, 1], где 1 это полное совпадение линии с точками, а -1 – максимальное расхождение. Благодаря этому, коэффициент может быть использован в качестве критерия подобия в неизменённом виде.

В рамках лабораторной работы, x и y – множества оценок для сравниваемых на текущей итерации алгоритма пользователей.

**Коэффициент Жаккара (Танимото):**

Является мерой подобия, или мерой близости. Определяет степень схожести некоторых множеств данных.

В общем виде, записывается как:

$$k = \frac{c}{a + b - c}$$

Формула 2.5 – коэффициент Жаккара

Где:

k – коэффициент в диапазоне [0, 1]

a – размер первого множества

b – размер второго множества

c – число общих элементов для множеств a и b

**Задание:** необходимо реализовать функции для расчёта коэффициентов подобия по евклидовой метрике и коэффициенту Пирсона, имеющие одинаковую сигнатуру (список параметров и возвращаемое значение).

Провести следующие исследования:

- сравнить результаты вычисления коэффициентов подобия при помощи евклидовой метрики и расстояния городских кварталов. Определить зависимость коэффициента подобия от числа совпадений и не совпадений оценок по одним и тем же фильмам.
- разработать функцию построения графиков подобных графику 2.2 для любой пары пользователей. Исследовать зависимость числового значения коэффициента Пирсона от отношения оценок.
- разработать функцию расчёта коэффициента Жаккара на основе информации о числе совпадений оценок, с учётом коэффициента подобия.

**Справка:** простейшая функция рисования графиков

```
import turtle

turtle.color("black")

turtle.penup()
turtle.goto(0,0)
turtle.pendown()
turtle.write(" Y")
turtle.goto(0,-100)
turtle.write("0")
turtle.right(90)
turtle.goto(100,-100)
turtle.write(" X")
turtle.penup()
```

```
turtle.goto(50,-50)
turtle.pendown()
turtle.circle(2)
turtle.write("hail")

turtle.done()
```

## Раздел 3: Подбор рекомендации

Очевидным способом подбора рекомендаций, является поиск пользователей с наибольшим коэффициентом подобия и рекомендация фильмов из их списка, которые не были просмотрены (оценены) вами.

С учётом зависимости от коэффициента подобия с пользователями, таблица ожидаемых оценок будет выглядеть следующим образом:

| user_id | k | оценка фильма 1 | оценка фильма 2 | оценка фильма 3 |
|---|---|---|---|---|
| 1 | 0,9 | 4 | 3 | 5 |
| 2 | 0,8 | 4 | | 4 |
| 3 | 0,7 | | 3 | 5 |
| 4 | 0,6 | 4 | 5 | |
| 5 | 0,5 | 5 | 4 | 3 |
| прогноз | | 4,17 | 3,6 | 4,5 |

Таблица 3.1 – таблица ожидаемых оценок по оценкам пользователей

Где ожидаемая оценка каждого фильма рассчитывается по формуле:

$$\text{предполагаемая оценка} = \frac{\sum(k * \text{оценка})}{\sum k}$$

Формула 3.1 – расчёт ожидаемого рейтинга

Значение ожидаемой оценки фильма может быть использовано для фильтрации результатов подбора рекомендации.

Другим способом подбора рекомендаций, является поиск подобных фильмов.

Методика определения коэффициента подобия фильмов аналогична методике определения коэффициента подобия для людей, достаточно лишь составить словари, где первого уровня ключами будут названия фильмов, а ключами второго уровня номера пользователей поставивших им оценку.

На основании таблицы подобия фильмов, можно реализовать систему рекомендаций, основанную на группах товаров.

Подобная система строится в два этапа:

1. Построение таблицы соответствия товаров. Для каждого товара вычисляется коэффициенты подобия с другими товарами, а затем, строится таблица максимально подобных товаров.
2. Просмотр списка товаров, которым пользователь выставил оценки и получение товаров подобных им.

Применительно к оценкам фильмов, таблица рекомендаций может выглядеть следующим образом:

| просмотренные | оценка | фильм_1 | фильм_2 | фильм_3 |
|---|---|---|---|---|
| фильм_1 | 4 | 0,8 | 0,3 | 0,5 |
| фильм_2 | 5 | 0,4 | 0,7 | 0,1 |
| фильм_3 | 3 | 0,1 | 0,5 | 0,8 |
| прогноз | | 4,2 | 4,1 | 3,5 |

Таблица 3.2 – таблица ожидаемых оценок по коэффициенту подобия фильмов

В первой колонке список фильмов, которым пользователь поставил оценки. С третьей колонки и далее, список фильмов подобных тем, которые указаны в первой колонке и их коэффициенты подобия.

Расчёт предполагаемой оценки фильмов выполняется по формуле 3.1

**Задание:** необходимо разработать и реализовать функции подбора списка рекомендаций фильмов для произвольного пользователя по оценке подобия пользователей и оценке подобия фильмов.

Провести следующие исследования:

- построить график подобия для набора фильмов, используя коэффициент Пирсона. Провести исследования значений коэффициента для разных сочетаний фильмов.
- измерить время построения таблиц подобия для обеих функций. Установить зависимость времени построения от размера набора данных.
- измерить время выработки рекомендаций для обеих функций. Предложить варианты оптимизации.

**Справка:** замер времени можно произвести, используя объект time

```
import  time
t = time.time()
```

## Итоговое задание:

Разработать и реализовать функцию выработки рекомендаций музыкальных композиций, на основании информации о числе прослушиваний и принадлежности к тому или иному музыкальному жанру для произвольного пользователя. (ссылка на необходимый набор данных указана на странице №1)

**Структура данных в архиве hetrec2011-lastfm-2k**:
artists.dat – содержит информацию об исполнителях `номер|название|ссылка`
user_artists.dat – содержит информацию о пользователях, артистах и числе прослушиваний
`пользователь|исполнитель|прослушивания`
tags.dat – содержит информацию о жанрах `номер|название`
user_taggedartists.dat – содержит информацию о пользователе, жанре и исполнителе
`номер пользователя|номер артиста|номер жанра`
Во всех документах, разделяющий символ - tab (“\t”)
Более подробную информацию о содержимом архива можно получить в файле Readme