

Описание результатов и алгоритмов работы программы.

Для того чтобы рекомендовать фильмы, каждому фильму нужно назначить среднюю оценку, выставленную всеми пользователями. Это делалось следующим образом: для каждой оценки данного фильма рассчитывался “вес” оценки (на сколько объективна оценка данного фильма сделанная конкретным пользователем) по формуле

$$w_i = \log(NG) \cdot \log(NM)$$

В данной формуле NG - число фильмов данного жанра, которые просмотрел пользователь.

NM - общее число фильмов, которое просмотрел пользователь.

Затем, выставлялась общая оценка фильма по формуле,

$$RatingProdWeight = \sum_i \frac{R_i}{5} \cdot w_i$$

где R_i - оценка фильма, выставленная одним пользователем, w_i - “вес” оценки.

Суммирование ведется по всем оценкам данного фильма.

Найдем любимые жанры пользователя. Для этого каждому жанру, который когда либо смотрел пользователь, поставим оценку по формуле

$$User Genres Rating = \frac{NG}{NM} \cdot R_{mean}$$

В данной формуле NG - число фильмов данного жанра, которые просмотрел пользователь.

NM - общее число фильмов, которое просмотрел пользователь.

R_{mean} - средняя оценка выставленная этим пользователем фильмам из данного жанра

Jupyter siriusAlproject Последняя контрольная точка: 8 минут назад (автосохранение)

Файл Редактировать Вид Вставка Ячейка Ядро Widgets Справка Не доверять Python 3 (ipykernel)

Ввод [22]:

```
#### Код для формирования рекомендаций для конкретного пользователя (с номером userID)

# Для какого-то пользователя userID делаем таблицу userpref с просмотренными Жанрами
# и считаем рейтинг Жанров пользователя UserGenresRating.
# Умножаем среднюю оценку фильмов каждого жанра на количество просмотренных фильмов данного жанра
# и делим на общее число просмотренных фильмов

userID=2345 # это ID пользователя, для которого будем делать рекомендации
NumberGenres = 5 #это число жанров пользователя, для которых будем рекомендовать фильмы

UserMovieCount=ratings[ratings['userId']==userID]['UserMovieCount'].iloc[0]
userpref=ratings[ratings['userId']==userID].groupby(['genres']).agg({'rating':'mean','userId':'count'})
userpref=userpref.rename(columns={"rating":"UserGenresMeanRating", "userId": "UserGenresCount"})
userpref['UserGenresRating']=(userpref['UserGenresCount']/UserMovieCount)*userpref['UserGenresMeanRating']

#Выбираем сколько-то NumberGenres первых по значению UserGenresRating жанров - Самые Любимые жанры
userpref=userpref.sort_values(by=['UserGenresRating'], ascending=False)[:NumberGenres]
userpref.head(10)
```

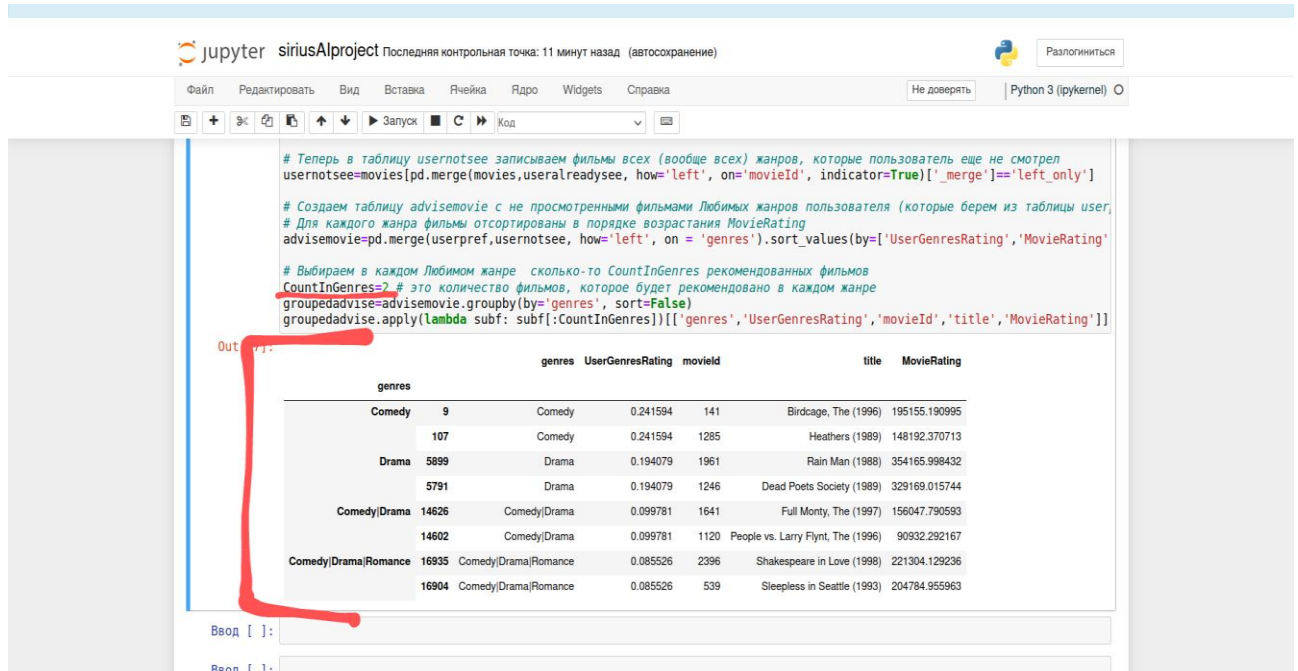
Out[22]:

genres	UserGenresMeanRating	UserGenresCount	UserGenresRating
Comedy	2.977477	111	0.241594
Drama	3.587838	74	0.194079
Comedy/Drama	3.412500	40	0.099781
Comedy/Drama/Romance	3.545455	33	0.085526
Comedy/Romance	3.085714	35	0.078947

Ввод [23]: # Находим ВСЕ фильмы для любимых жанров пользователя, которые он еще не видел

1. Выбираем пользователя, для которого будем составлять рекомендации. Выбираем из сколько любимых жанров пользователя мы будем рекомендовать ему фильмы. В результате получаем следующую табличку: пять жанров, колонна, в которой стоит число, которое оценивает степень интереса пользователя к жанру.
2. Для выбранных любимых жанров пользователя мы выбираем все фильмы, которые он не смотрел.

При помощи переменной `CountInGenres` (у нас она равна 2) указываем, какое число лучших фильмов из ранее выбранных жанров рекомендовать (два фильма для каждого жанра).



The screenshot shows a Jupyter Notebook interface with the following components:

- Header:** "jupyter siriusAIproject" with a status bar indicating "Последняя контрольная точка: 11 минут назад (автосохранение)".
- Menu Bar:** Файл, Редактировать, Вид, Вставка, Ячейка, Ядро, Widgets, Справка.
- Toolbar:** Includes icons for file operations, a "Запуск" (Run) button, and a "Python 3 (ipykernel)" label.
- Code Cell:** Contains the following Python code:

```
# Теперь в таблицу usernotsee записываем фильмы всех (вообще всех) жанров, которые пользователь еще не смотрел
usernotsee=movies[pd.merge(movies,useralreadysee, how='left', on='movieId', indicator=True)['_merge']=='left_only']

# Создаем таблицу advisemovie с не просмотренными фильмами Любимых жанров пользователя (которые берем из таблицы user)
# Для каждого жанра фильмы отсортированы в порядке возрастания MovieRating
advisemovie=pd.merge(userpref,usernotsee, how='left', on = 'genres').sort_values(by=['UserGenresRating', 'MovieRating'])

# Выбираем в каждом Любимом жанре сколько-то CountInGenres рекомендованных фильмов
CountInGenres=2 # это количество фильмов, которое будет рекомендовано в каждом жанре
groupedadvise=advisemovie.groupby(by='genres', sort=False)
groupedadvise.apply(lambda subf: subf[:CountInGenres])[['genres', 'UserGenresRating', 'movieId', 'title', 'MovieRating']]
```
- Output:** A table with 6 columns: genres, UserGenresRating, movieId, title, and MovieRating. The table lists movie recommendations grouped by genre. A red bracket highlights the first two rows of the "Comedy" group.

genres	UserGenresRating	movieId	title	MovieRating
Comedy	0.241594	141	Birdcage, The (1996)	195155.190995
Comedy	0.241594	1285	Heathers (1989)	148192.370713
Drama	0.194079	1961	Rain Man (1988)	354165.998432
Drama	0.194079	1246	Dead Poets Society (1989)	329169.015744
Comedy/Drama	0.099781	1641	Full Monty, The (1997)	156047.790593
Comedy/Drama	0.099781	1120	People vs. Larry Flynt, The (1996)	90932.282167
Comedy/Drama/Romance	0.085526	2396	Shakespeare in Love (1998)	221304.129236
Comedy/Drama/Romance	0.085526	539	Sleepless in Seattle (1993)	204784.955963

В исходных данных есть пользователи, которые посмотрели более 5000 фильмов, а это почти 14 лет просмотра одного нового фильма каждый день. Таких странных пользователей мы удалили, чтобы их оценки не учитывать в рейтинге фильмов.