Лабораторная работа 7. Рекомендательная система

Задание

На основе предложенного датасета и инструкции, приведенной ниже, построить на языке Python простейшую рекомендательную систему.
После выполнения основного задания в соответствии с инструкцией, заменить исходные фильмы на два любых других из предложенного датасета и привести результаты выборки.
Сформировать текстовый отчет, в котором отобразить основные шаги выполнения работы, привести скриншоты построенных графиков (гистограмм) и полученных результатов, а также результирующий код программы. Помимо отчета приложить к заданию исходник программы (*.py или *.ipynb, либо zip архив с исходниками проекта).

Датасет

В качестве датасета использовать https://grouplens.org/datasets/movielens/

Строим рекомендательную систему

Первым делом импортируем pandas и numpy.

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Далее загружаем датасет при помощи метода pandas `read_csv()` . Датасет разделен табами (не запятыми!), поэтому используем `\t` как `sep` параметр. Названия столбуцоы можно получить при помощи параметра `names`.

```
df = pd.read_csv('u.data', sep='\t',
names=['user_id','item_id','rating','titmestamp'])
```

Теперь давайте проверим заголовок данных, чтобы увидеть данные, с которыми мы имеем дело

```
df.head()
```

Было бы хорошо, если бы мы могли видеть названия фильмов вместо того, чтобы иметь дело с идентификаторами. Давайте загрузим названия фильмов и объединим их с этим набором данных.

```
movie_titles = pd.read_csv('Movie_Titles')movie_titles.head()
```

Поскольку столбцы `item_id` одинаковы, мы можем объединить эти наборы данных в этом столбце.

```
df = pd.merge(df, movie_titles, on='item_id')df.head()
```

Посмотрим, что содержат столбцы:

- `user_id` - идентификатор пользователя, который оценил фильм.
- `item_id` - идентификатор фильма.
- `rating` - Рейтинг, который пользователь дал фильму, между 1 и 5.
- `timestamp` - время, когда фильм был оценен.
- `title` - название фильма.

Используя команды `describe` или `info` мы можем получить краткое описание нашего набора данных. Это важно для того, чтобы мы могли понять набор данных, с которым мы работаем.

```
df.describe()
```

Теперь давайте создадим фрейм данных со средней оценкой для каждого фильма и количеством оценок. Мы собираемся использовать эти рейтинги для вычисления корреляции между фильмами позже. Фильмы с высоким коэффициентом корреляции - это фильмы, которые наиболее похожи друг на друга. В нашем случае мы будем использовать коэффициент корреляции Пирсона. Это число будет лежать между -1 и 1. 1 указывает на положительную линейную корреляцию, а -1 указывает на отрицательную корреляцию. 0 указывает на отсутствие линейной корреляции. Поэтому фильмы с нулевой корреляцией совсем не похожи. Для создания этого фрейма данных мы используем функциональность pandas `groupby`. Мы группируем набор данных по столбцу `title` и вычисляем его среднее значение, чтобы получить среднюю оценку для каждого фильма.

```
ratings =
pd.DataFrame(df.groupby('title')['rating'].mean())ratings.head()
```

Далее нам хотелось бы увидеть количество оценок для каждого фильма. Мы делаем это путем создания столбца number_of_ratings. Это важно, чтобы мы могли видеть взаимосвязь между средней оценкой фильма и количеством оценок, которые получил фильм. Вполне возможно, что 5-звездочный фильм был оценен только одним человеком. Поэтому статистически неверно классифицировать этот фильм как 5-звездочный фильм. Поэтому нам нужно будет установить порог для минимального количества рейтингов при построении системы рекомендаций. Для создания этого нового столбца мы используем утилиту pandas `groupby`. Мы группируемся по столбцу заголовка и затем используем функцию подсчета, чтобы

вычислить количество оценок каждого фильма. После этого мы просматриваем новый фрейм данных с помощью функции `head()`.

```
ratings['number_of_ratings'] =
df.groupby('title')['rating'].count()
ratings.head()
```

Давайте теперь построим гистограмму, используя функцию построения графиков pandas, чтобы визуализировать распределение оценок.

```
import matplotlib.pyplot as plt
%matplotlib inline
ratings['rating'].hist(bins=50)
```

Далее давайте визуализируем столбец `number_of_ratings` таким же образом.

```
ratings['number_of_ratings'].hist(bins=60)
```

Из приведенной выше гистограммы видно, что большинство фильмов имеют несколько рейтингов. Фильмы с наибольшим рейтингом - это те, которые наиболее известны.

Давайте теперь проверим взаимосвязь между рейтингом фильма и количеством оценок. Мы делаем это путем построения точечной диаграммы с использованием Seaborn. Seaborn позволяет нам сделать это с помощью функции `jointplot()`.

```
import seaborn as sns
sns.jointplot(x='rating', y='number_of_ratings', data=ratings)
```

Из диаграммы видно, что это положительная связь между средней оценкой фильма и количеством оценок. График показывает, что чем больше оценок получает фильм, тем выше средняя оценка. Это важно отметить, особенно при выборе порогового значения для количества оценок на фильм.

Давайте теперь создадим простую систему рекомендаций на основе предметов. Для этого нам нужно преобразовать наш набор данных в матрицу с заголовками фильмов в качестве столбцов, `user_id` в качестве индекса и рейтингами в качестве значений. Сделав это, мы получим фрейм данных со столбцами в качестве заголовков фильмов и строками в качестве идентификаторов пользователей. Каждый столбец представляет все оценки фильма всеми пользователями. Рейтинг отображается как NAN, где пользователь не оценил определенный фильм. Мы будем использовать эту матрицу для вычисления корреляции между оценками одного фильма и остальными фильмами в матрице. Для создания матрицы фильма мы используем утилиту pandas `pivot_table`.

```
movie_matrix = df.pivot_table(index='user_id', columns='title',
values='rating')
movie_matrix.head()
```

Теперь давайте посмотрим на фильмы с самым высоким рейтингом и выберем два из них для работы в этой простой системе рекомендаций. Мы используем утилиту pandas sort_values и устанавливаем значение по возрастанию в false, чтобы упорядочить фильмы из самых рейтинговых. Затем мы используем функцию `head()` для просмотра 10 лучших.

```
ratings.sort_values('number_of_ratings', ascending=False).head(10)
```

Предположим, что пользователь смотрел Air Force One (1997) и Contact (1997). Мы хотели бы рекомендовать фильмы этому пользователю на основе этой истории просмотра. Цель состоит в том, чтобы найти фильмы, похожие на Contact (1997) и Air Force One (1997), которые мы будем рекомендовать этому пользователю. Мы можем добиться этого, рассчитав корреляцию между рейтингами этих двух фильмов и оценками остальных фильмов в наборе данных. Первым шагом является создание кадра данных с рейтингами этих фильмов из нашего `movie_matrix`.

```
AFO_user_rating = movie_matrix['Air Force One (1997)']
contact_user_rating = movie_matrix['Contact (1997)']
```

Теперь у нас есть данные, показывающие `user_id` и рейтинг, который они дали двум фильмам. Давайте посмотрим на них ниже.

```
AFO_user_rating.head()contact_user_rating.head()
```

Чтобы вычислить корреляцию между двумя фреймами данных, мы используем функциональность Pandas Corrwith. `Corrwith` вычисляет попарную корреляцию строк или столбцов двух объектов данных. Давайте использовать эту функциональность, чтобы получить корреляцию между рейтингом каждого фильма и рейтингами фильма Air Force One.

```
similar_to_air_force_one=movie_matrix.corrwith(AFO_user_rating)
similar_to_air_force_one.head()
```

Эти же действия повторим для фильма Contact.

```
similar_to_contact = movie_matrix.corrwith(contact_user_rating)
similar_to_contact.head()
```

Как отмечалось ранее, в нашей матрице было очень много пропущенных значений, поскольку не все фильмы были оценены всеми пользователями. Поэтому мы отбрасываем эти нулевые значения и преобразуем результаты корреляции в кадры данных, чтобы результаты выглядели более привлекательными.

```
corr_contact = pd.DataFrame(similar_to_contact,
columns=['Correlation'])
corr_contact.dropna(inplace=True)
corr_contact.head()corr_AFO =
pd.DataFrame(similar_to_air_force_one, columns=['correlation'])
corr_AFO.dropna(inplace=True)
corr_AFO.head()
```

Эти два кадра данных выше показывают нам фильмы, которые наиболее похожи на фильмы «Contact» (1997) и «Air Force One» (1997) соответственно. Однако у нас есть проблема в том, что некоторые фильмы имеют очень мало оценок и могут в конечном итоге быть рекомендованы просто потому, что один или два человека дали им 5-звездочный рейтинг. Мы можем исправить это, установив порог для количества оценок. Из гистограммы ранее мы видели резкое снижение количества рейтингов со 100. Поэтому мы установим это в качестве порога, однако это число, с которым вы можете играть, пока не получите подходящий вариант. Для этого нам нужно объединить два кадра данных со столбцом `number_of_ratings` в кадре оценок.

```
corr_AFO = corr_AFO.join(ratings['number_of_ratings'])
corr_contact =
corr_contact.join(ratings['number_of_ratings'])corr_AFO
.head()corr_contact.head()
```

Теперь мы получим фильмы, которые наиболее похожи на Air Force One (1997), ограничив их фильмами, которые имеют не менее 100 рецензий. Затем мы сортируем их по столбцу корреляции и просматриваем первые 10.

```
corr_AFO[corr_AFO['number_of_ratings'] >
100].sort_values(by='correlation', ascending=False).head(10)
```

То же самое сделаем для второго фильма.

```
corr_contact[corr_contact['number_of_ratings'] >
100].sort_values(by='Correlation', ascending=False).head(10)
```