A. Рекомендательная система

Есть некоторый новостной сайт, на котором представлены разные статьи. Сейчас здесь нет какой-то серьезной рекомендательной системы, пользователю показываются все недавние статьи по популярности.

В вашем распоряжении имеются данные о том, какие статьи (item\_id) открывал пользователь (user\_id) и порядок (order), в котором они были открыты. Порядок начинается с 11 (самое раннее действие) и увеличивается со временем.

[Обучающая выборка](https://assets.codeforces.com/rounds/338595/c510ad677402d8bd/train.csv) представлена тремя колонками [user\_id, item\_id, order], где user\_id показывает идентификатор пользователя, item\_id идентификатор страницы и order — порядковый номер посещенной страницы.

user\_id,item\_id,order  
0,0,1  
0,1,2  
0,2,3

Кроме того, в вашем распоряжении имеются [описания пользователя](https://assets.codeforces.com/rounds/338595/c510ad677402d8bd/user-features.csv). Колонки 0-15 — обезличенная информация о пользователе, колонка user\_id — ключ, по которому можно объединить данные по просмотрам и фичи пользователя. На основе предоставленных данных, вам необходимо предсказать следующие 3 сайта, которые посетит каждый пользователь из [тестового выборки](https://assets.codeforces.com/rounds/338595/c510ad677402d8bd/test.csv). Аналогично тренировочным данным, порядок в ней начинается с 1 и увеличивается со временем: order=1 означает, что это следующий посещенный сайт после окончания последовательности в обучающей выборке.

user\_id,order  
0,1  
0,2  
0,3

Файл [sample-submission.csv](https://assets.codeforces.com/rounds/338595/c510ad677402d8bd/sample-submission.csv) содержит пример решения для загрузки в тестовую систему. В вашем сабмите должны быть колонки с пользователем user\_id, предсказаниями айтема для этого пользователя item\_id и колонка order, описывающая порядок айтемов, в котором пользователь их увидит. Не забывайте, что порядок предсказаний в сабмите должен начинаться с 1 и увеличиваться с каждым следующим предсказанием.

user\_id,item\_id,order  
0,1,1  
0,1,2  
0,1,3

В качестве метрики для оценивания качества используется метрика ранжирования mean average precision at k (MAP@k) при k=3. То есть в метрике будут участвовать только 3 первых предсказания. На лидерборде показывается метрика умноженная на 10000, то есть результат будет выглядеть как 10000 \* MAP@k.

Расчет данной метрики можно представить в три этапа:

1. Считаем precision at k:

Precision@k=1k∑i=0k(yitrue==yipred)Precision@k=1k∑i=0k(ytruei==ypredi)

2. Считаем average precision at k:

AP@k=1k∑i=0k(yitrue==yipred)∗Precision@iAP@k=1k∑i=0k(ytruei==ypredi)∗Precision@i

3. Берем среднее от average precision at k:

MAP@k=1N∑i=0NAP@kMAP@k=1N∑i=0NAP@k

Чтобы реализовать данную функцию локально можно использовать следующий алгоритм.

На вход нам приходят два pd.DataFrame gt и preds с колонками [user\_id, item\_id, order]. gt – ground truth, то есть настоящие данные, а preds – наши предсказания. Тогда мы можем использовать следующие шаги для расчета метрики.

1. Отфильтровать gt и preds по колонке order, чтобы его значение было не больше k (порядка k в MAP@k).

2. Сджойнить два датафрейма по user\_id и order. Здесь нам важно сохранить информацию о том, какой item\_id на каком месте был для каждого пользователя.

После этих шагов мы готовы к расчету метрик по трем вышеописанным шагам для расчета MAP@k

3. Добавим в объединенный датафрейм (joined) колонку is\_right, показывающую, что на текущем месте ответ был правильным. 4. Так как для расчета precision@k нам нужно знать количество всех правильных событий до текущего момента, то посчитаем кумулятивную сумму колонки is\_right в разрезе по пользователям. Назовем ее is\_right\_cum.

5. Теперь посчитаем precision@k поделив колонку is\_right\_cum на order. То есть разделим количество правильно угаданных событий на количество всех событий до текущего момента.

6. В AP@k участвуют только те precision@k, для которых в текущий момент времени был правильно угадан ответ. Поэтому мы можем добавить еще одну колонку p@k\_masked, которая является перемножением p@k на is\_right.

7. Посчитаем среднее в разрезе каждого пользователя по колонке p@k\_masked, получим AP@k для каждого пользователя.

8. Теперь возьмем глобальное среднее по AP@k и получим итоговую метрику MAP@k.

Ниже представлен пример имплементации данного алгоритма, который можно использовать для локальной валидации.

**def** calc\_map\_k(gt: pd.**DataFrame**, preds: pd.**DataFrame**, k=3) -> **float**:  
 **# filter first k elements of ground truth**  
 gt = gt[gt.order <= k]  
 **# filter first k elements of predictions**  
 preds = preds[preds.order <= k]  
 **# join ground truth and predictions by user\_id and order, fill missed values from predictions by some non-existent value**  
 joined = gt.merge(preds, how=**"left"**, on=[**'user\_id'**, **"order"**]).fillna(-12345)  
 **# create indicator of right predictions**  
 joined[**"is\_right"**] = (joined.item\_id\_x == joined.item\_id\_y).astype(**int**)  
 **# calculate cumulative sum of all right predictions before current order**  
 joined[**"is\_right\_cum"**] = joined.groupby(**"user\_id"**).is\_right.cumsum()  
 **# normalize it by order (precision@k)**  
 joined[**"p@k"**] = joined[**"is\_right\_cum"**] / joined[**"order"**]  
 **# add relevance mask**  
 joined[**"p@k\_masked"**] = joined[**"p@k"**] \* joined[**"is\_right"**]  
 **# calculate mean user based (average precision @ k)**  
 ap = joined.groupby(**"user\_id"**)[**"p@k\_masked"**].mean()  
 **# calculate mean average precision @ k**  
 **return** ap.mean()