



# Your First RecSys

**Даниил Потапов**

Руководитель группы персонализации и рекомендательных систем  
MTC BigData





# Валидация, метрики и бейзлайны



# План

- Валидация
- Метрики
- Бейзлайны



# Валидация

Основная цель валидации - оценить качество модели перед её использованием

Поэтому процесс валидации должен *максимально точно воспроизводить условия, в которых модель будет использоваться.*



# Валидация

Что хотим от модели?

- Предсказанное значение для пары пользователь-объект
- Ранжирование объектов для пользователя

Как будет использоваться модель?

- Рекомендации будут считаться раз в какой-то период
- Онлайн-рекомендации

Какие технические ограничения?

- Время на обучение и время на построение рекомендаций
- Доступность данных в онлайн-режиме



# Валидация

Какие есть особенности у задачи?

- Рекомендуем ли для пользователя объекты, с которым он уже взаимодействовал
  - В контентных системах (книги, фильмы и тд) скорее всего нет
  - В продуктивном ритейле, наоборот, вероятнее всего да
- Cold start - как много появится пользователей или объектов, для которых не известна история по взаимодействиям
- Как много взаимодействий по пользователям есть в данных

# Валидация

Подходы для разбиения данных на train/test:

- Случайное
- В хронологическом порядке
- По временным периодам

Original Data

x	x		x	
		x		x
x		x		
	x			x
		x	x	



Training Set

x	x		MASK	
		x		MASK
MASK		x		
	MASK			x
		x	MASK	

Test Set

			x	
				x
x				
	x			
			x	



# Валидация

## Случайное разбиение

- Leave one out
  - Для test оставляем одно случайное взаимодействие
- Leave P out
  - Для test оставляем P случайных взаимодействий

## Методы

- `from sklearn.model_selection import train_test_split, LeaveOneOut, LeavePOut`
- `pandas.DataFrame.sample`
- `pandas.core.groupby.DataFrameGroupBy.sample`





# Валидация

Хронологическое разбиение - оставляем в test только последние взаимодействия по пользователям или объектам

- Last one out
- Last P out

Подход

1. `pandas.DataFrame.sort_values(by='date_column')`
2. `pandas.core.groupby.DataFrameGroupBy`
  - `last/tail`
  - `first/head`
  - `nth`



# Валидация

Разбиение по времени - оставляем в test только конкретный временной промежуток, например 1 день или 1 неделя.

## Пример

1. Выбрать две даты
  - `train_max_date` - ограничение для train
  - `test_max_date` - ограничение для test
2. На основе дат выделить train/test
  - `train = df[df['date_column'] < train_max_date]`
  - `test = df[(df['date_column'] >= train_max_date) & (df['date_column'] < test_max_date)]`



# Валидация

При генерации надо учитывать также *cold start* и *warm start* сценарии:

- Cold start - это пользователи и объекты из test, для которых не известны взаимодействия
- Warm start - это пользователи и объекты из test, которых не было в train, но для которых на момент построения рекомендаций известны взаимодействия



# Метрики

Метрики в рекомендательных системах можно разделить на следующие группы

- Регрессионные
- Классификационные
- Ранжирующие



# Метрики

Правильные метки мы можем представить в виде `pandas.DataFrame` со следующими столбцами:

- `user_id` - ID пользователя
- `item_id` - ID объекта
- `value` - оценка взаимодействия
  - опционально, может и не быть, если мы, например, за взаимодействие берем просто клик или факт покупки

Соответственно, рекомендации/предсказания в следующем:

- `user_id` - ID пользователя
- `item_id` - ID объекта
- `value` - предсказанная оценка взаимодействия
  - Численная оценка, например, рейтинг
  - Позиция, если мы ранжируем контент



# Метрики

Чтобы соотнести правильные ответы с предсказаниями, мы можем их объединить через left join

	user_id	item_id
0	Аня	Мастер и Маргарита
1	Боря	451° по Фаренгейту
2	Вася	Зеленая миля
3	Вася	Рита Хейуорт и спасение из Шоушенка

	user_id	item_id	rank
0	Аня	Отверженные	1
1	Аня	Двенадцать стульев	2
2	Аня	Герои нашего времени	3
3	Боря	451° по Фаренгейту	1
4	Боря	1984	2
5	Боря	О дивный новый мир	3
6	Вася	Десять негритят	1
7	Вася	Искра жизни	2
8	Вася	Зеленая миля	3

		rank
user_id	item_id	
Аня	Мастер и Маргарита	NaN
Боря	451° по Фаренгейту	1.0
Вася	Зеленая миля	3.0
	Рита Хейуорт и спасение из Шоушенка	NaN

```
df_merged = df_true\  
    .set_index(['user_id', 'item_id'])  
    .join(  
        df_recs.set_index(['user_id', 'item_id']),  
        how='left')
```



# Метрики

Регрессионные метрики применяются для оценки качества предсказанных моделью значений


- Mean Absolute Error  $MAE = \frac{1}{N} \sum_{i=1}^N |y_{true} - y_{pred}|$
- Mean Squared Error  $MSE = \frac{1}{N} \sum_{i=1}^N (y_{true} - y_{pred})^2$
- Rooted Mean Squared Error  $RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_{true} - y_{pred})^2}$



# Метрики

	user_id	item_id	value
0	Аня	Мастер и Маргарита	4
1	Боря	451° по Фаренгейту	5
2	Вася	Зеленая миля	3
3	Вася	Рита Хейуорт и спасение из Шоушенка	5

	user_id	item_id	value
0	Аня	Мастер и Маргарита	3.28
1	Боря	451° по Фаренгейту	3.50
2	Вася	Зеленая миля	4.06
3	Вася	Рита Хейуорт и спасение из Шоушенка	4.73



		value_true	value_recs	MAE	MSE
user_id	item_id				
Аня	Мастер и Маргарита	4	3.28	0.72	0.5184
Боря	451° по Фаренгейту	5	3.50	1.50	2.2500
Вася	Зеленая миля	3	4.06	1.06	1.1236
	Рита Хейуорт и спасение из Шоушенка	5	4.73	0.27	0.0729





# Метрики

Классификационные метрики оценивают качество топ-N рекомендаций с точки зрения бинарной классификации. Все считается на основе 4 базовых случаев:

- True positive (TP) - модель рекомендовала объект, с которым пользователь провзаимодействовал
- False positive (FP) - модель рекомендовала объект, с которым пользователь не провзаимодействовал
- True negative (TN) - модель не рекомендовала объект, с которым пользователь не провзаимодействовал
- False negative (FN) - модель не рекомендовала объект, с которым пользователь провзаимодействовал

# Метрики

		True condition			
Total population		Condition positive	Condition negative	Prevalence = $\frac{\sum \text{Condition positive}}{\sum \text{Total population}}$	Accuracy (ACC) = $\frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$
Predicted condition	Predicted condition positive	True positive	False positive, Type I error	Positive predictive value (PPV), Precision = $\frac{\sum \text{True positive}}{\sum \text{Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\sum \text{False positive}}{\sum \text{Predicted condition positive}}$
	Predicted condition negative	False negative, Type II error	True negative	False omission rate (FOR) = $\frac{\sum \text{False negative}}{\sum \text{Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\sum \text{True negative}}{\sum \text{Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR}^+}{\text{LR}^-}$  $F_1 \text{ score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate = $\frac{\sum \text{False negative}}{\sum \text{Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\sum \text{True negative}}{\sum \text{Condition negative}}$	Negative likelihood ratio (LR-) = $\frac{\text{FNR}}{\text{TNR}}$	



# Метрики

Наиболее популярным метриками являются

- Precision@K
  - Формула:  $TP / (TP + FP)$
  - Можно заметить, что под positives мы понимаем рекомендованные объекты, то есть наш топ-K, значит  $TP + FP = K$
  - Итоговая формула:  $TP / K$
  - Интерпретируется как доля релевантных рекомендаций
- Recall@K
  - Формула:  $TP / (TP + FN)$
  - $TP + FN$  это количество известных релевантных объектов для пользователя
  - Интерпретируется как доля релевантных объектов, попавших в рекомендации



# Метрики

	user_id	item_id
0	Аня	Мастер и Маргарита
1	Боря	451° по Фаренгейту
2	Вася	Зеленая миля
3	Вася	Рита Хейуорт и спасение из Шоушенка

	user_id	item_id	rank
0	Аня	Отверженные	1
1	Аня	Двенадцать стульев	2
2	Аня	Герои нашего времени	3
3	Боря	451° по Фаренгейту	1
4	Боря	1984	2
5	Боря	О дивный новый мир	3
6	Вася	Десять негритят	1
7	Вася	Искра жизни	2
8	Вася	Зеленая миля	3

		rank
user_id	item_id	
Аня	Мастер и Маргарита	NaN
Боря	451° по Фаренгейту	1.0
Вася	Зеленая миля	3.0
	Рита Хейуорт и спасение из Шоушенка	NaN



# Метрики

		rank
user_id	item_id	
Аня	Мастер и Маргарита	NaN
Боря	451° по Фаренгейту	1.0
Вася	Зеленая миля	3.0
	Рита Хейуорт и спасение из Шоушенка	NaN



		rank	hit@2	hit@2/2
user_id	item_id			
Аня	Мастер и Маргарита	NaN	False	0.0
Боря	451° по Фаренгейту	1.0	True	0.5
Вася	Зеленая миля	3.0	False	0.0
	Рита Хейуорт и спасение из Шоушенка	NaN	False	0.0

Получение True positives для k=2

```
df_merged['hit@2'] = df_merged['rank'] <= 2  
df_merged['hit@2/2'] = df_merged['hit@2'] / 2
```

Получение Precision@2

```
df_merged.groupby(level=0)['hit@2/2'].sum().mean() == 0.1666
```

Но groupby по user\_id делать не обязательно

```
users_count = df_merged.index.get_level_values('user_id').nunique()  
df_merged['hit@2/2'].sum() / users_count == 0.1666
```



# Метрики

Классификационные метрики уже неплохо показывают качество наших топ-К рекомендаций, но они учитывают только попадания. А мы также хотим, чтобы наши релевантные рекомендации находились как можно выше.

Здесь нам и помогут ранжирующие метрики, которые будут оценивать наши попадания, но с весами:

- Mean Reciprocal Rank
- Mean Average Precision
- Normalized Discounted Cumulative Gain



# Метрики

Mean Reciprocal Rank - средний обратный ранг

$$MRR = \frac{1}{N} \sum_{i=1}^N \frac{1}{rank_i}$$

$N$  - кол-во пользователей,  $rank_i$  - позиция первой релевантной рекомендации для пользователя  $i$

Причем если для пользователя мы не рекомендовали ничего релевантного, то дробь  $\frac{1}{rank_i}$  зануляется



# Метрики

Mean Average Precision - средняя точность по пользователям

$$MAP@K = \frac{1}{N} \sum_{i=1}^N AP@K(user_i)$$

$$AP@K(user) = \frac{1}{c_{user}} \sum_{i=1}^K Precision@i * rel_i$$

$N$  - кол-во пользователей

$c_{user}$  - кол-во релевантных объектов у пользователя

$rel_i$  - релевантность  $i$ -ой рекомендации





# Метрики

Normalized Discounted Cumulative Gain - взвешенная точность по пользователям

$$CG@K = \sum_{i=1}^K rel_i \quad rel_i - \text{вес } i\text{-ой позиции, 0 если не релевантна}$$

$$DCG@K = \sum_{i=1}^K \frac{rel_i}{\log_2(i+1)} \quad IDC@K = \sum_{i=1}^{\min(|Rel|, K)} \frac{rel_i}{\log_2(i+1)} \quad |Rel| - \text{кол-во релевантных объектов для пользователя}$$

$$NDCG@K = \frac{DCG@K}{IDC@K}$$



# Бейзлайны

Самый популярный бейзлайн - построить популярное :)

Обычно, гиперпараметром у такой модели является глубина истории, за которую считается популярность объектов

Также можно добавлять дополнительные фичи, по которым учитывать популярное:

- Популярное по возрастным или гендерным группам
- Популярное по жанрам или категориям
- Если данные последовательны, то можно считать популярность относительно предыдущего объекта, с которым пользователь взаимодействовал



# Бейзлайны

```
import pandas as pd
from itertools import islice, cycle

class PopularRecommender():
    def __init__(self, max_K=100, days=30, item_column='item_id', dt_column='date'):
        self.max_K = max_K
        self.days = days
        self.item_column = item_column
        self.dt_column = dt_column
        self.recommendations = []

    def fit(self, df):
        min_date = df[self.dt_column].max().normalize() - pd.DateOffset(days=self.days)
        dt_mask = df[self.dt_column] > min_date
        self.recommendations = df.loc[dt_mask, self.item_column].value_counts().head(self.max_K).index.values

    def recommend(self, users=None, N=10):
        recs = self.recommendations[:N]
        if users is None:
            return recs
        else:
            return list(islice(cycle([recs]), len(users)))
```



Попрактиковаться самим можно здесь:

<https://www.kaggle.com/sharhz23/metrics-validation-strategies-and-baselines>

Вопросы можно задавать как на самом kaggle, так и в

 [Telegram чате курса](#)

 sharthZ23