Recommender Systems

Lecture 3

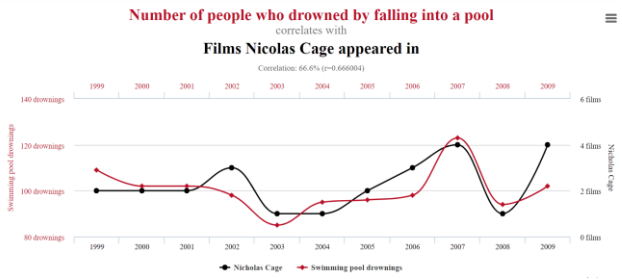
Previous lecture – simple recsys models

- Non-personalized recommendations
  - Popularity-based
- Content-based recommendations
  - Simple regression models:

$$\|\boldsymbol{y} - X\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|^2 \to \min$$

$$\sum_{k=1}^{n} \log\left(1 + e^{-y_k \cdot \boldsymbol{x}_k^{\mathsf{T}}\boldsymbol{w}}\right) + \lambda\|\boldsymbol{w}\|^2 \to \min$$

  - Similarity-based approach



Previous lecture - evaluation

- Naïve evaluation:
  - Prediction error / classification accuracy

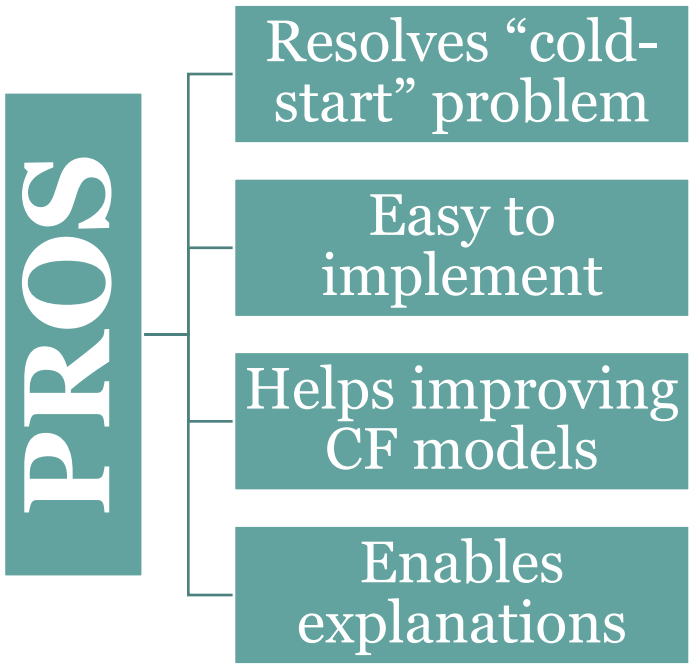
- A more user-oriented evaluation:
  - proximity to user's likes/dislikes

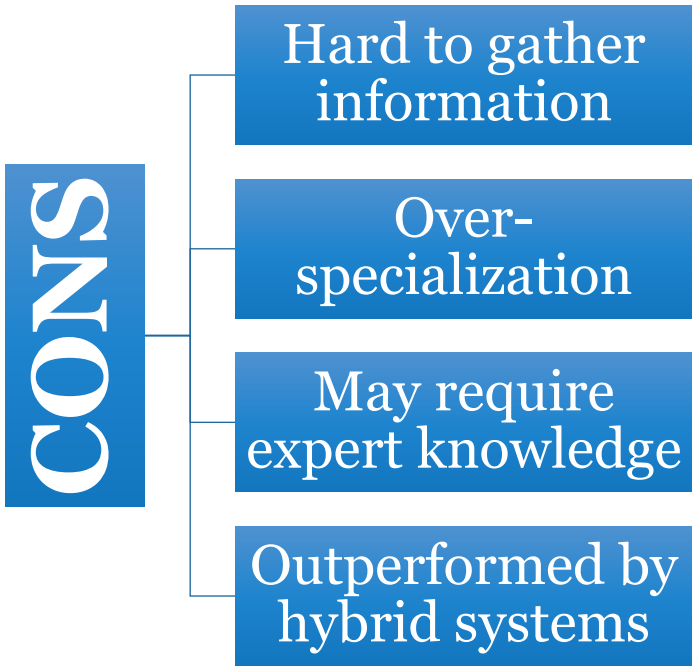
What can be used for content-based learning

Other options

- SVM
- Decision Tree-based models
  - CatBoost
  - LightGBM
  - XGBoost
  - Automated feature selection, e.g. as in LightAutoML
- Neural Networks

Content-based approach summary

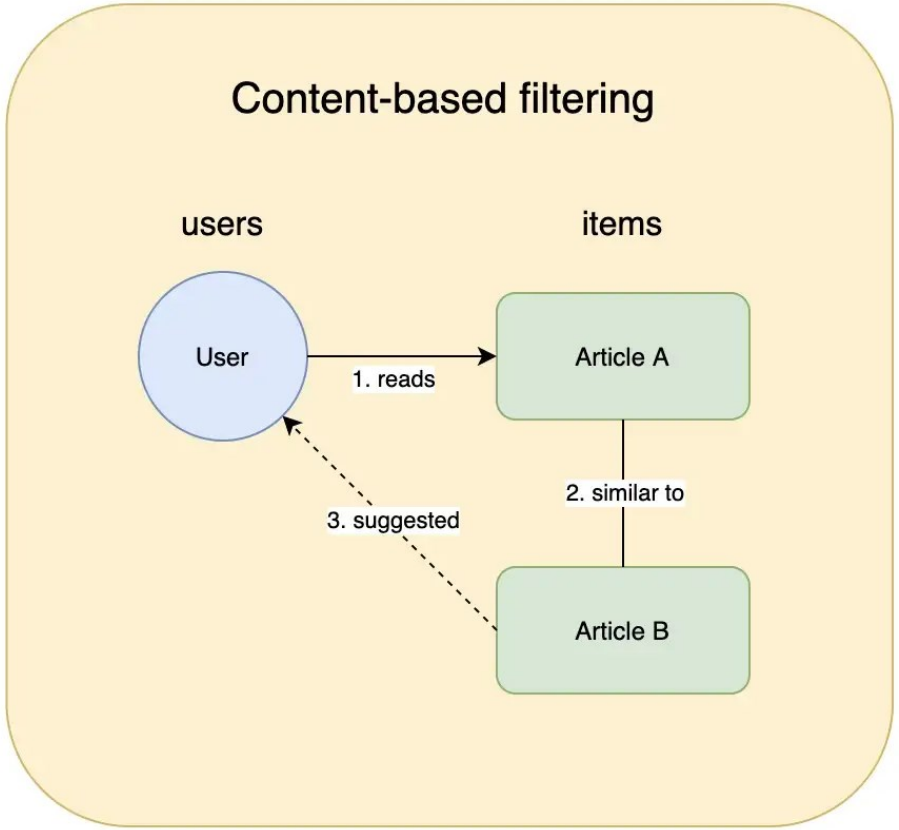


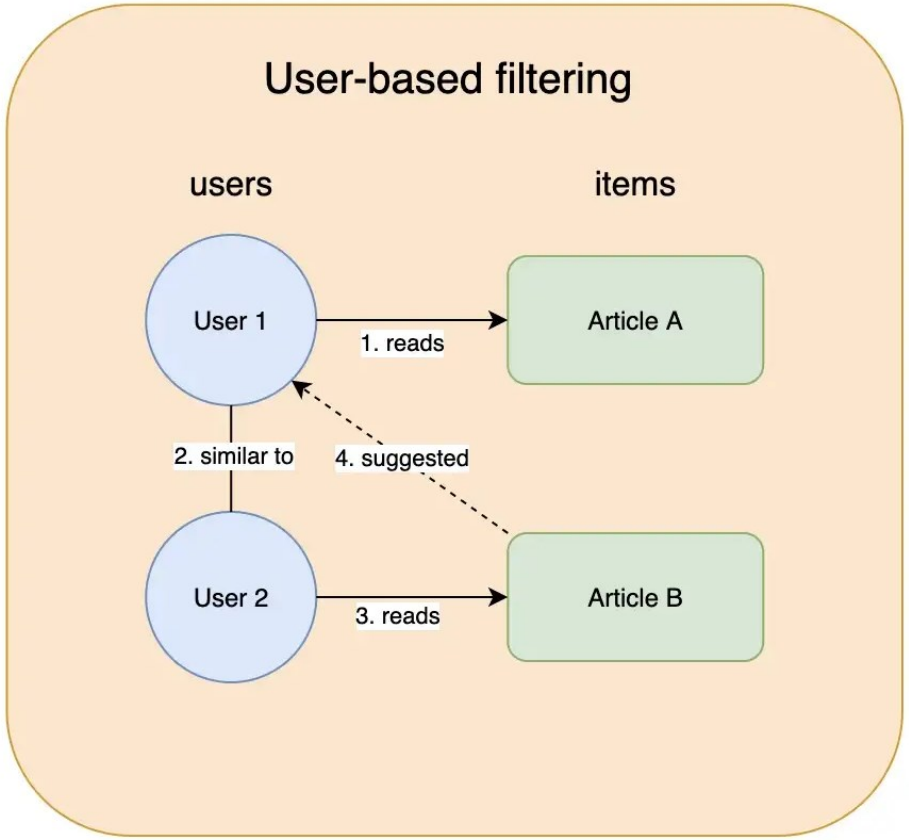


That’s where **deep learning** can really help: **features embedding.**

Better in hybrid approaches.

User-based filtering

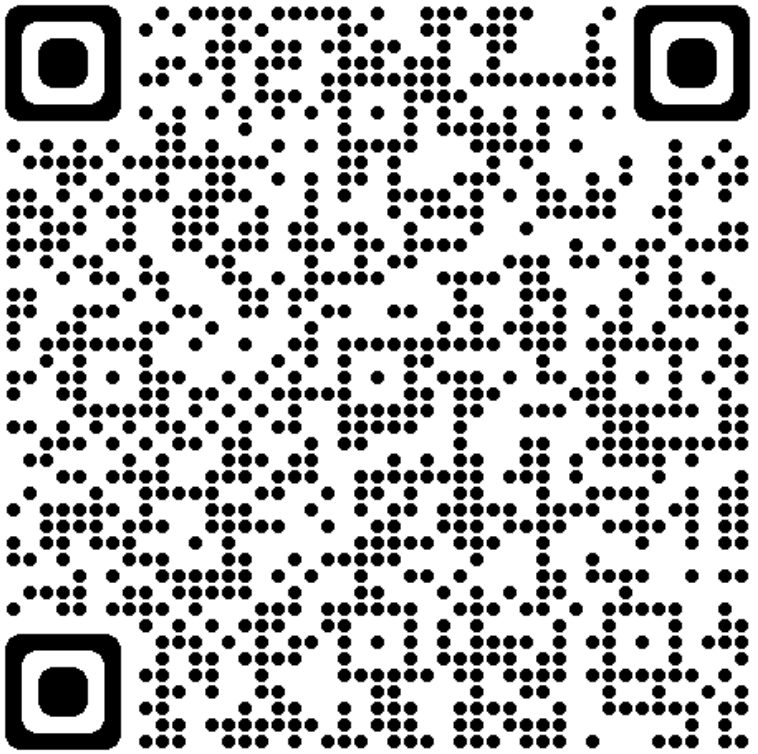




- still content-based filtering
- relevant scenario: user is a target (“cold” user)

Image credit: https://medium.com/nlplanet/two-minutes-nlp-tips-for-recommender-systems-with-nlp-a3f03555d65a

Lecture 2 Recap Quiz



https://forms.gle/9maGDqB14WQMoDNi7

Today's lecture

- Non-personalized recommendations
  - Baseline predictors
- Offline evaluation
  - Data preparation
  - Methodologies
  - Metrics

Baseline predictors

Rating Prediction Tasks

- Same rating from different users may have very different meanings.
- How can we capture systematic tendencies in rating patterns?

$$b_{ij} = \overline{r_{i:}}$$

$$b_{ij} = \overline{r_{:j}}$$

$$b_{ij} = \left(\overline{r_{i:}} + \overline{r_{:j}}\right) / 2$$

$$b_{ij} = \mu + g_i + f_j$$

Baseline predictors (Biases)

$$r_{ij} \approx b_{ij} = g_i + f_j + \mu$$

g_i – ***g****enerosity* of user i, i.e. tendency to assign higher or lower rating

f_j – ***f****avoredness of* item j, i.e. how likely it's to be praised or critiqued

μ – global average

tends to capture much of the observed signal

Baseline predictors - decoupled form

$$\begin{cases} g_i = \dfrac{1}{|I_i| + \lambda_i} \displaystyle\sum_{j \in I_i} (r_{ij} - \mu) \\ f_j = \dfrac{1}{|U_j| + \lambda_j} \displaystyle\sum_{i \in U_j} (r_{ij} - g_i - \mu) \end{cases}$$

I_i - set of items, rated by user i,
U_j - set of users, who rated items j
λ_i, λ_j - "damping" factors (can set to constant, e.g 25)

Ekstrand, M.D., Riedl, J.T. and Konstan, J.A., 2011. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, *4*(2), pp.81-173.

Baseline predictors via optimization

$$\text{argmin} \sum_{i,j \in O} (r_{ij} - g_i - f_j - \mu)^2 + \lambda \left(\sum_i g_i^2 + \sum_j f_j^2 \right)$$

O - is a set of all known entries (observed data)

Iterative alternating scheme:

$$\begin{cases} g_i = \dfrac{1}{|I_i| + \lambda} \displaystyle\sum_{j \in I_i} (r_{ij} - f_j - \mu) \\ f_j = \dfrac{1}{|U_j| + \lambda} \displaystyle\sum_{i \in U_j} (r_{ij} - g_i - \mu) \end{cases}$$

Can be optimized with gradient-based methods as well.

Baseline estimation – alternative scheme

$$\operatorname{argmin} \sum_{i,j \in O} (r_{ij} - g_i - f_j)^2 = L$$

$$\begin{cases} \frac{\partial L}{\partial f_i} = 0 \\ \frac{\partial L}{\partial f_j} = 0 \end{cases}$$

Baseline estimation – alternative scheme

$$\text{argmin} \sum_{i,j \in O} (r_{ij} - g_i - f_j)^2$$

$$\begin{cases} D_g \boldsymbol{g} + S\boldsymbol{f} = \boldsymbol{b}_g \\ D_f \boldsymbol{f} + S^\top \boldsymbol{g} = \boldsymbol{b}_f \end{cases}$$

S – sparsity pattern of the data
$D_g = \text{diag}(S\boldsymbol{e})$ – diagonal matrix of row nnz
$D_f = \text{diag}(S^\top \boldsymbol{e})$ – diagonal matrix of column nnz
$\boldsymbol{b}_g = R\boldsymbol{e}$ – row-aggregated ratings
$\boldsymbol{b}_f = R^\top \boldsymbol{e}$ – columns-aggregated ratings

$$\left(D_f - S^\top D_g^{-1} S\right)\boldsymbol{f} = \boldsymbol{b}_f - S^\top D_g^{-1} \boldsymbol{b}_g$$

Can be effectively solved with GMRES

kernel of the system - ?

additional constraint: $\boldsymbol{f} \perp \boldsymbol{e}$

Orthogonality constraint gives natural condition: $\sum_j f_j = 0$

$(f, e) = 0$

$(f, e) = \mu$

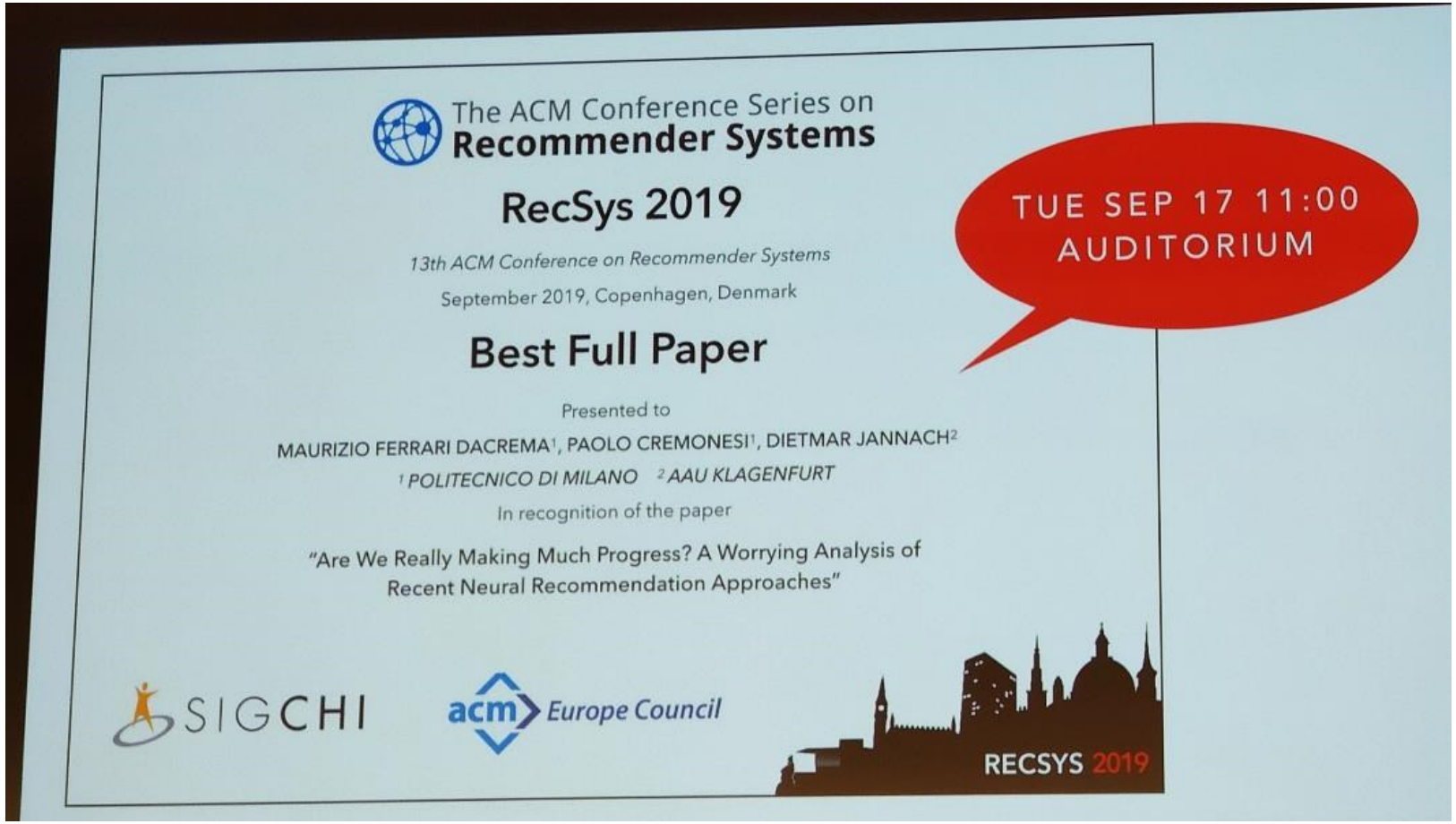
$f_i := f_i - \mu$

Evaluation of Recommender Systems

Demo

Simple recommendation engine in 3 lines of python code.

RecSys best Paper Award 2019



Evaluation of recommender systems

- Previously, we used the notion of “proximity” to evaluate the accuracy of recommendations.
  - What are the potential drawbacks of this approach?

- Suggest a different way of evaluating the quality of recommendations.

Types of evaluation

Offline Evaluation

- straightforward
- doesn't require real users
- can be used to benchmark the generalizability of models
- may not correlate with online/business metrics

Online Evaluation

- real users of a commercial recsys
- resembles clinical trials
- no public access, limited to specific domain and data

User studies

- users are recruited for an evaluation task
- can be performed by both scientists and practitioners
- can be biased

Factors influencing quality of recommendations

- Length of user profile
  - active users vs bots
  - “warm” vs “hot” users
- Concentration of popular items in user history
  - short head vs long tail items
- p-core filtering

“…if the algorithm discards the most popular or unpopular items, they will never appear as recommendations, and customers who have purchased only those items will not get recommendation.”
From Amazon paper

p-core filtering effects

Table 1: Statistics of datasets.

Dataset		ML-1M	Lastfm	Yelp	Epinions	Book-X	AMZe
origin	#User	6,038	1,892	1,326,101	22,164	105,283	4,201,696
	#Item	3,533	17,632	174,567	296,277	340,556	476,002
	#Record	575,281	92,834	5,261,669	922,267	1,149,780	7,824,482
	Density	2.697e-2	2.783e-3	2.273e-5	1.404e-4	3.207e-5	3.912e-6
5-filter	#User	6,034	1,874	227,109	21,995	22,072	253,994
	#Item	3,125	2,828	123,985	31,678	43,748	145,199
	#Record	574,376	71,411	3,419,587	550,117	623,405	2,109,869
	Density	3.046e-2	1.348e-2	1.214e-4	7.895e-4	6.456e-4	5.721e-5
10-filter	#User	5,950	1,867	96,168	21,111	12,720	63,161
	#Item	2,811	1,530	80,351	14,030	18,318	85,930
	#Record	571,549	62,984	2,458,153	434,162	443,196	949,416
	Density	3.412e-2	2.205e-2	3.181e-4	1.466e-3	1.902e-3	1.749e-4
Timestamp		√	×	√	√	×	√

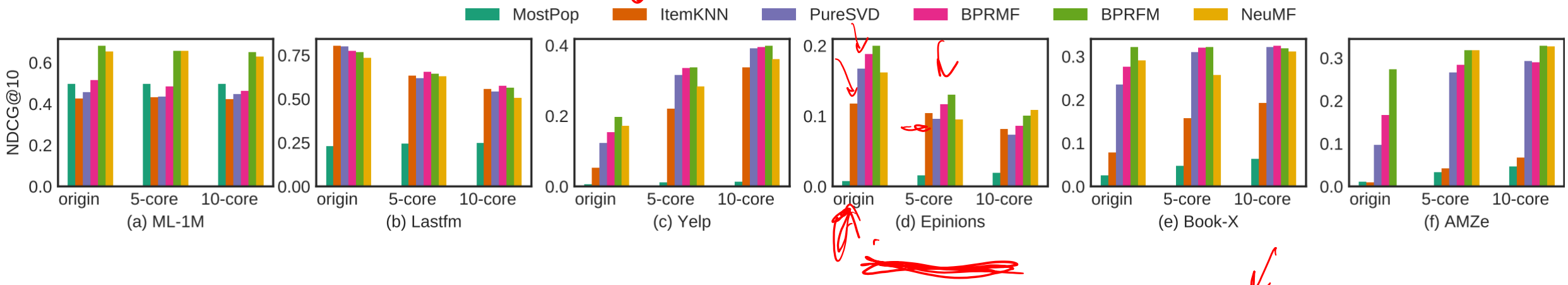


Image source: Sun, Zhu, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. "Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison." In *Fourteenth ACM conference on recommender systems*, pp. 23-32. 2020.

Evaluation methodologies in offline

Can’t we just use standard ML procedures?

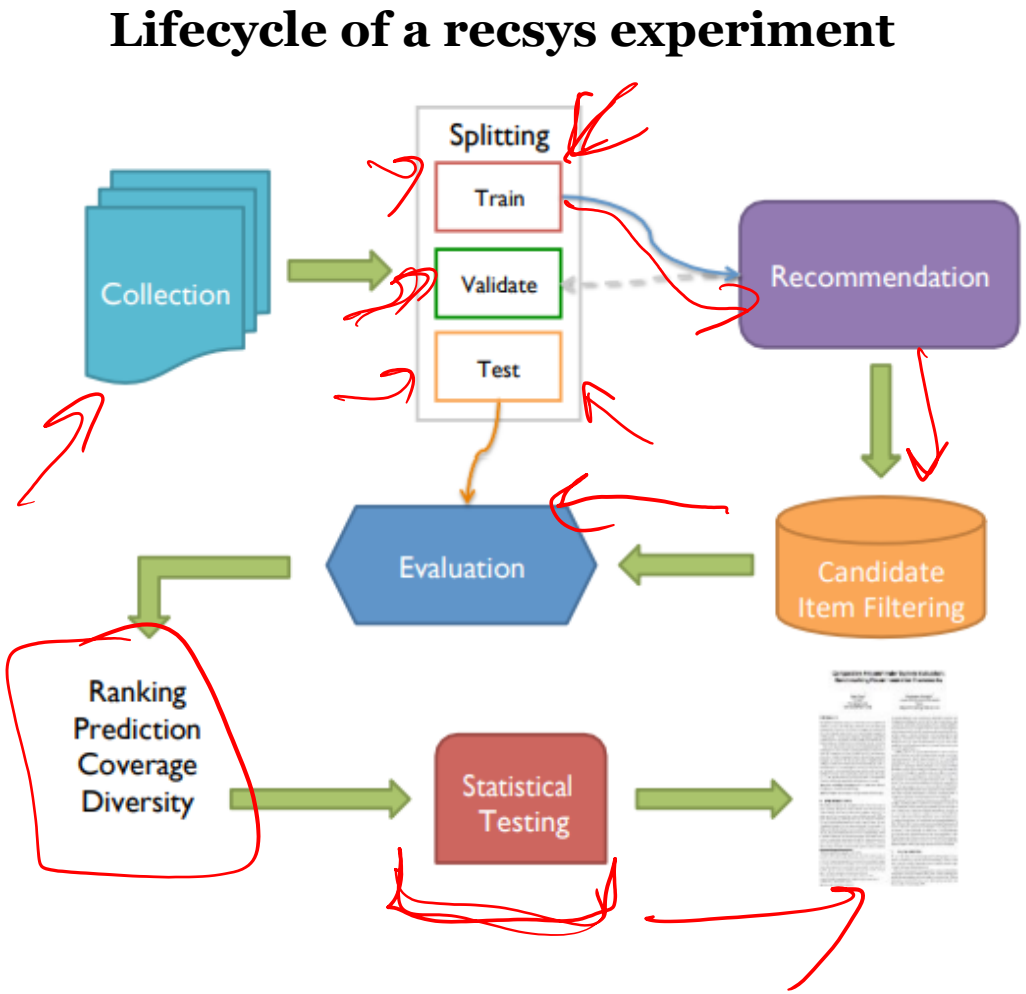
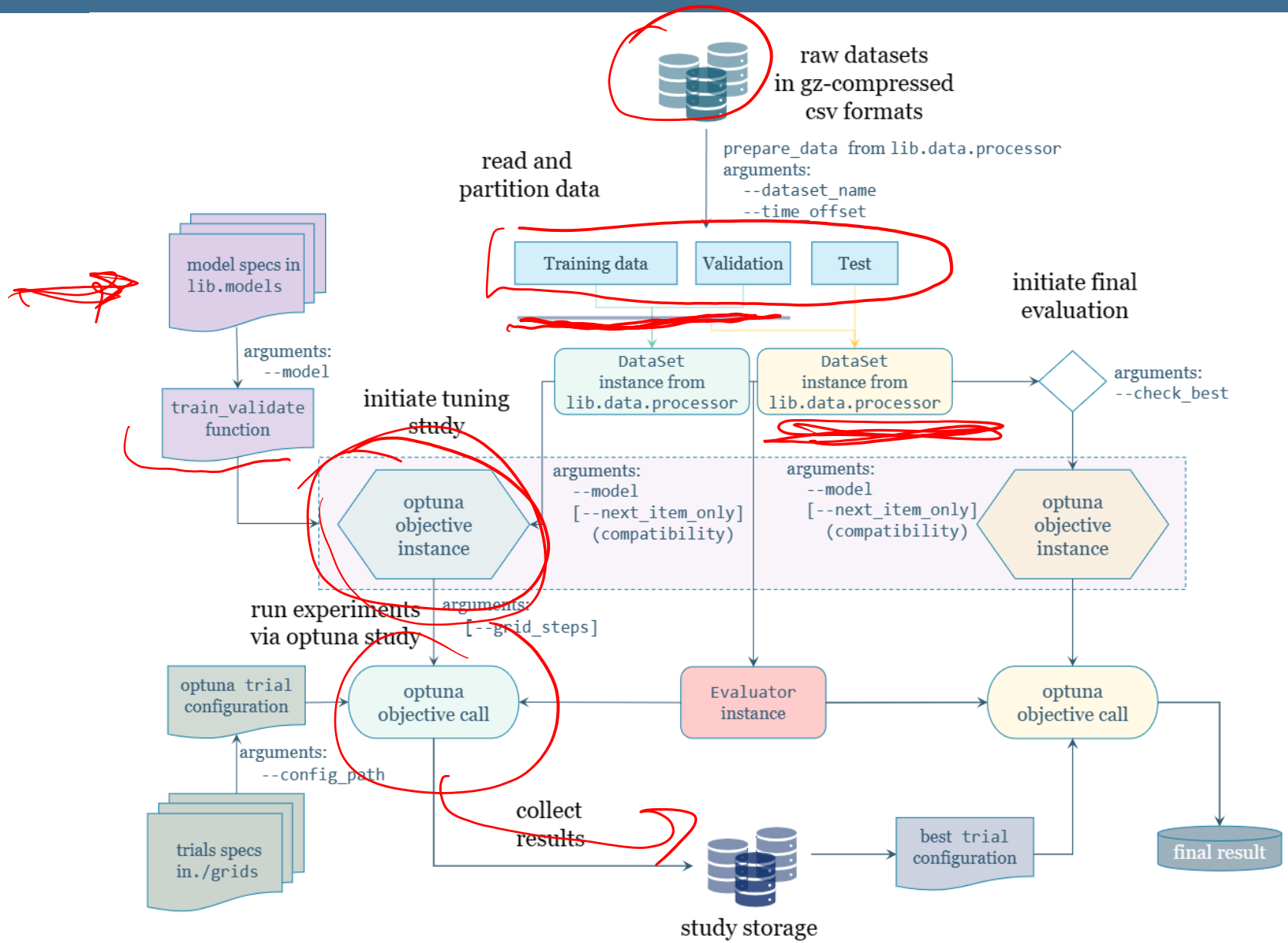
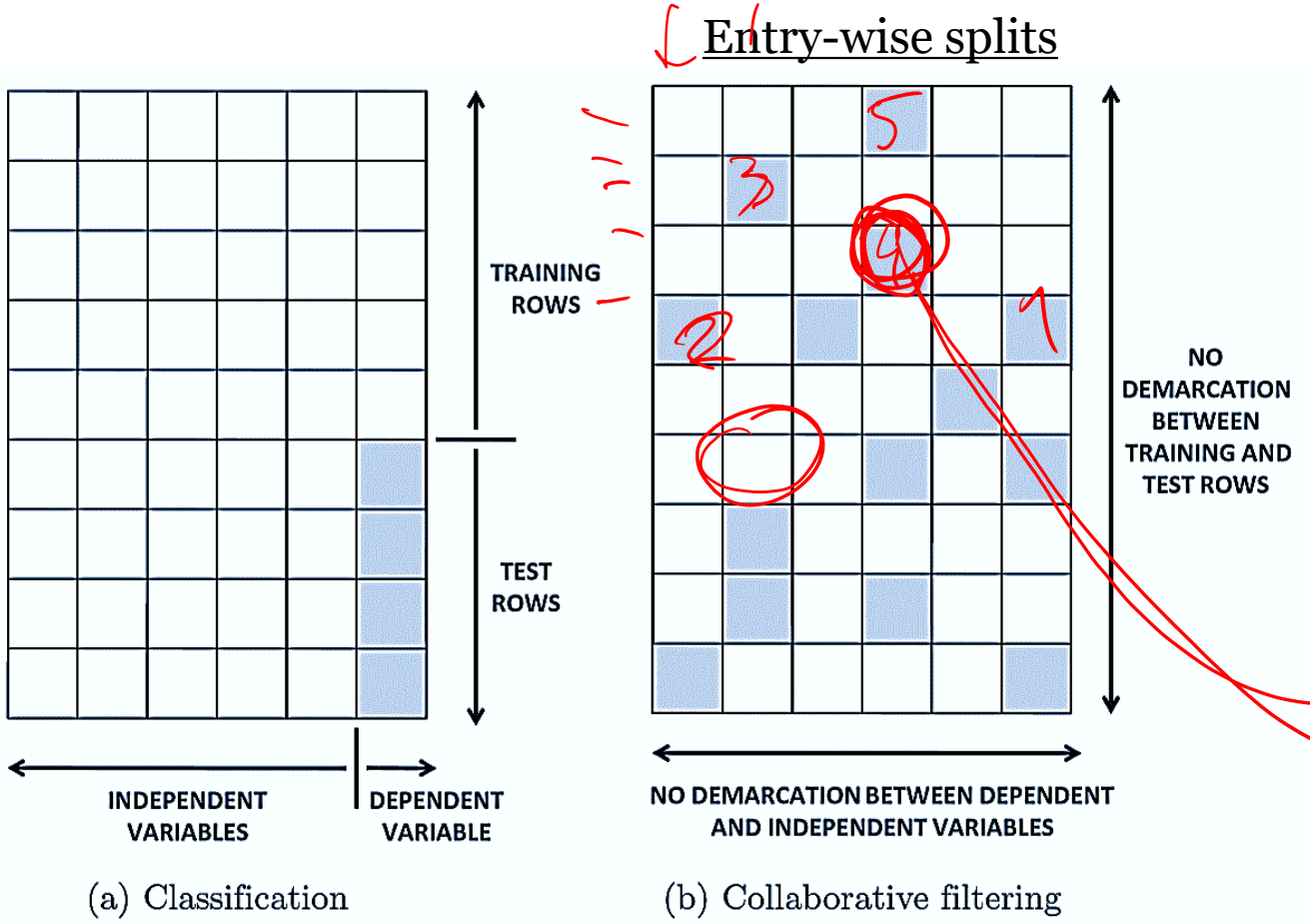


Image source: Bellogín, Alejandro, and Alan Said. "*Improving accountability in recommender systems research through reproducibility*." User Modeling and User-Adapted Interaction (2021): 1-37.

Experimentation pipeline example



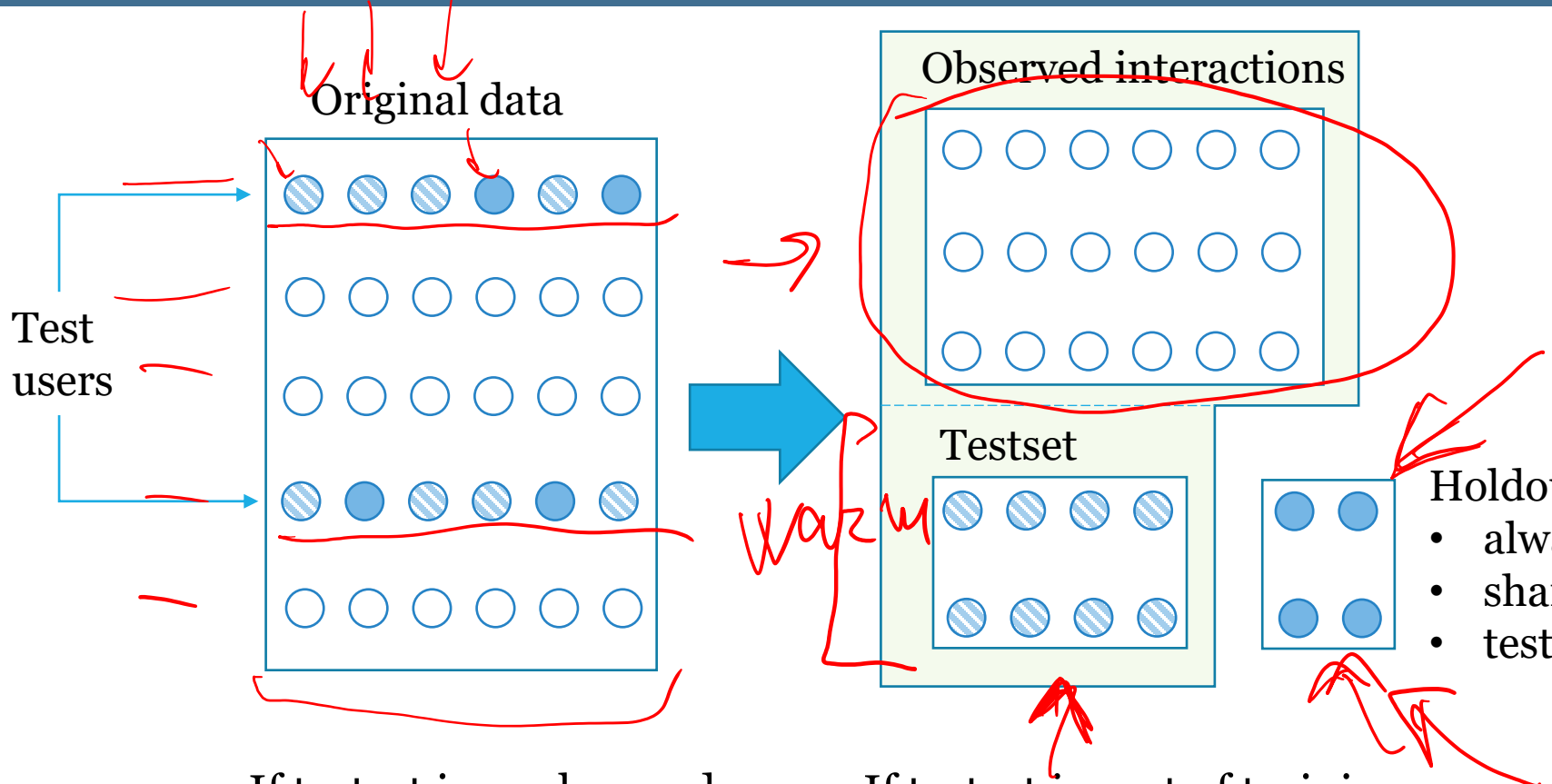
Classification/Regression or Recommendation?



- generating recommendations is often viewed as a separate task
- can still be formulated as a regression/classification problem
  - but dealing with new entities (users, items) requires special efforts

Image credit: "Recommender Systems. The Textbook", Charu C. Aggarwal, 2016

User-wise data split schemes



Holdout items sampling

Strategies

- Random
- Rating-based (e.g., top-rated)
- Temporal (e.g., most recent)

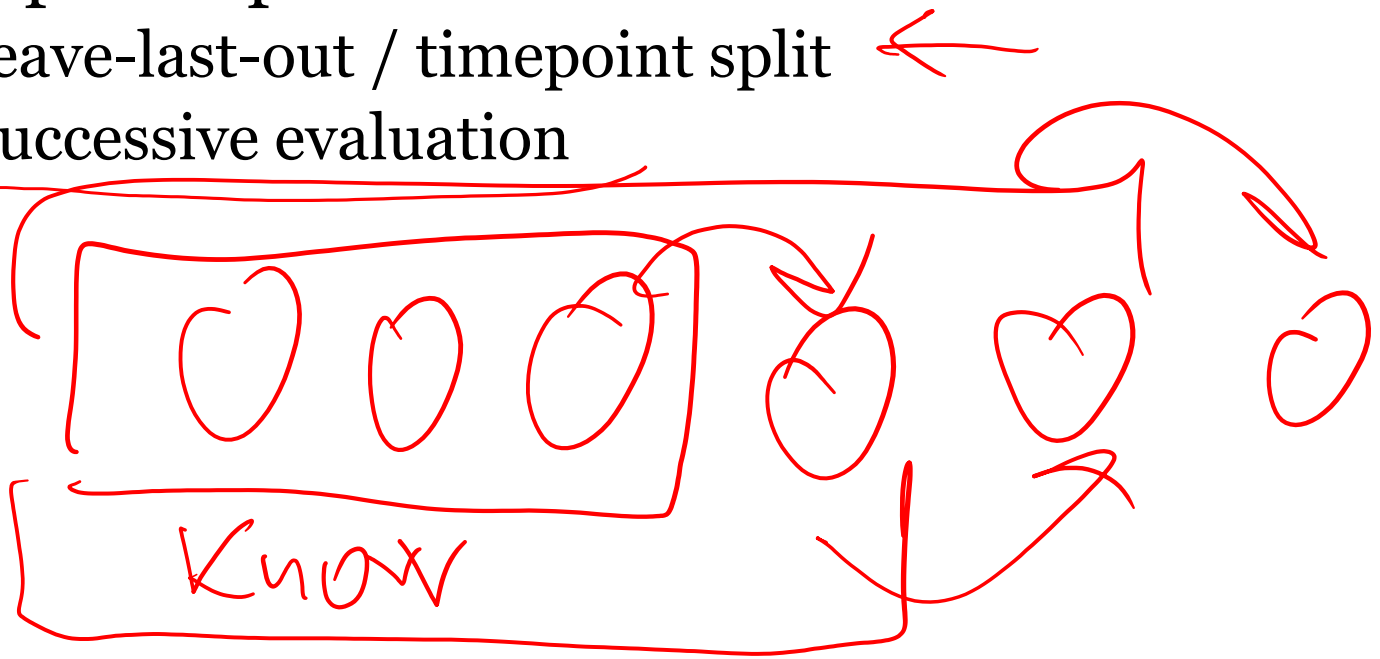
Sample size

- Fixed number of items (e.g., 1)
- Fixed percentage of items

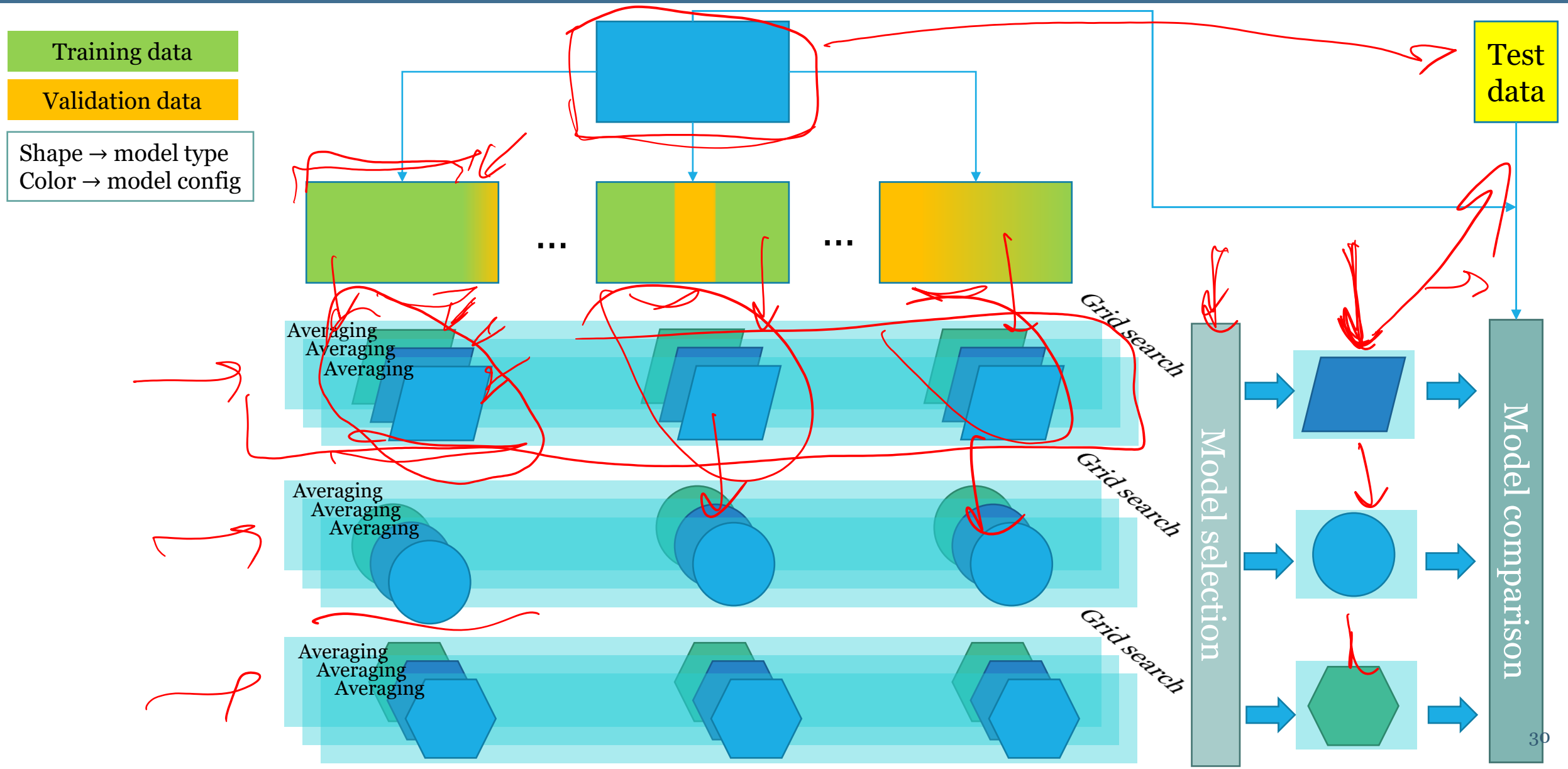
Typical data splitting strategies

- random holdout split
  - leave-%-out or leave-m-out, $m = 1, 2, \ldots$
  - use k-fold cross-validation

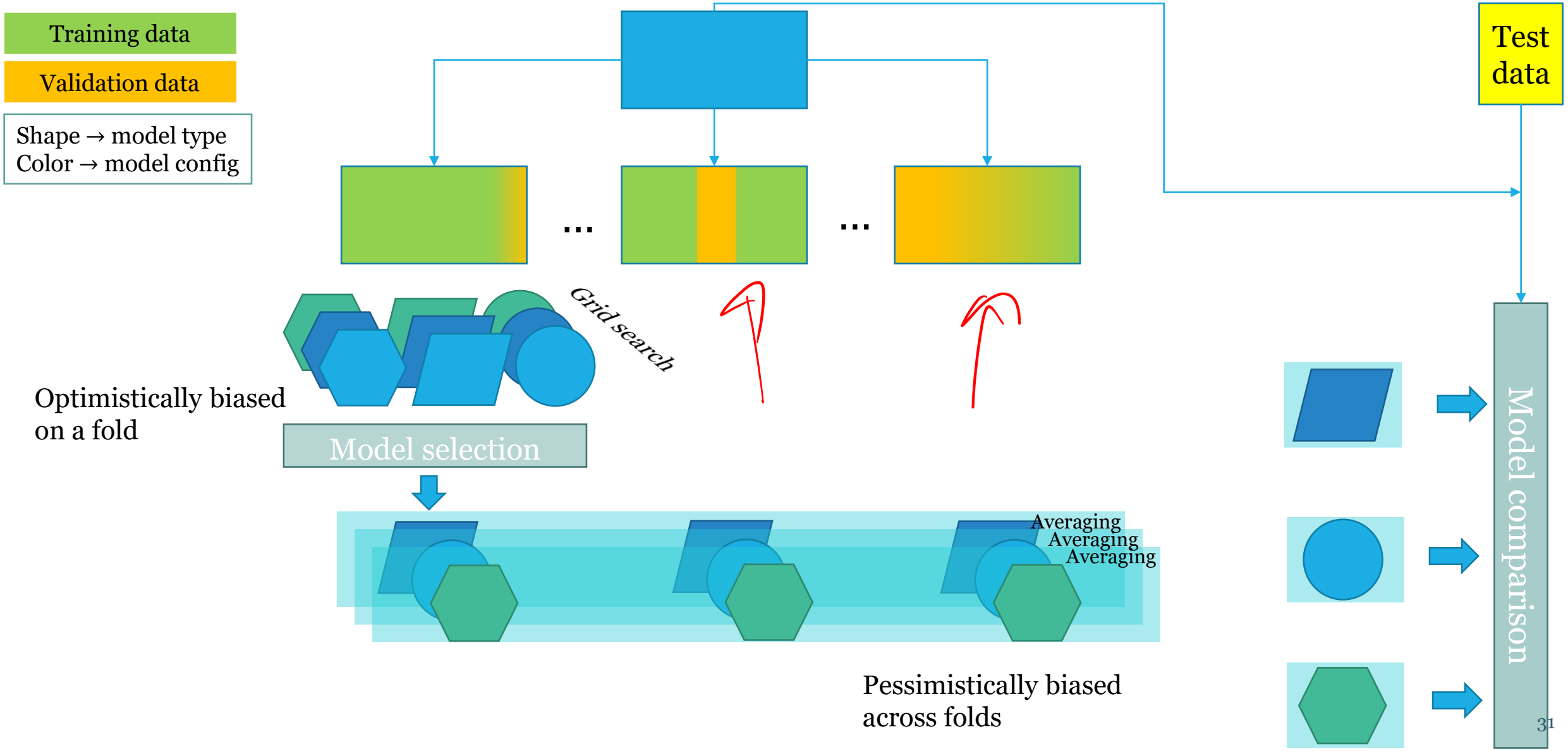
- temporal split
  - leave-last-out / timepoint split
  - successive evaluation



k-fold cross-validation process

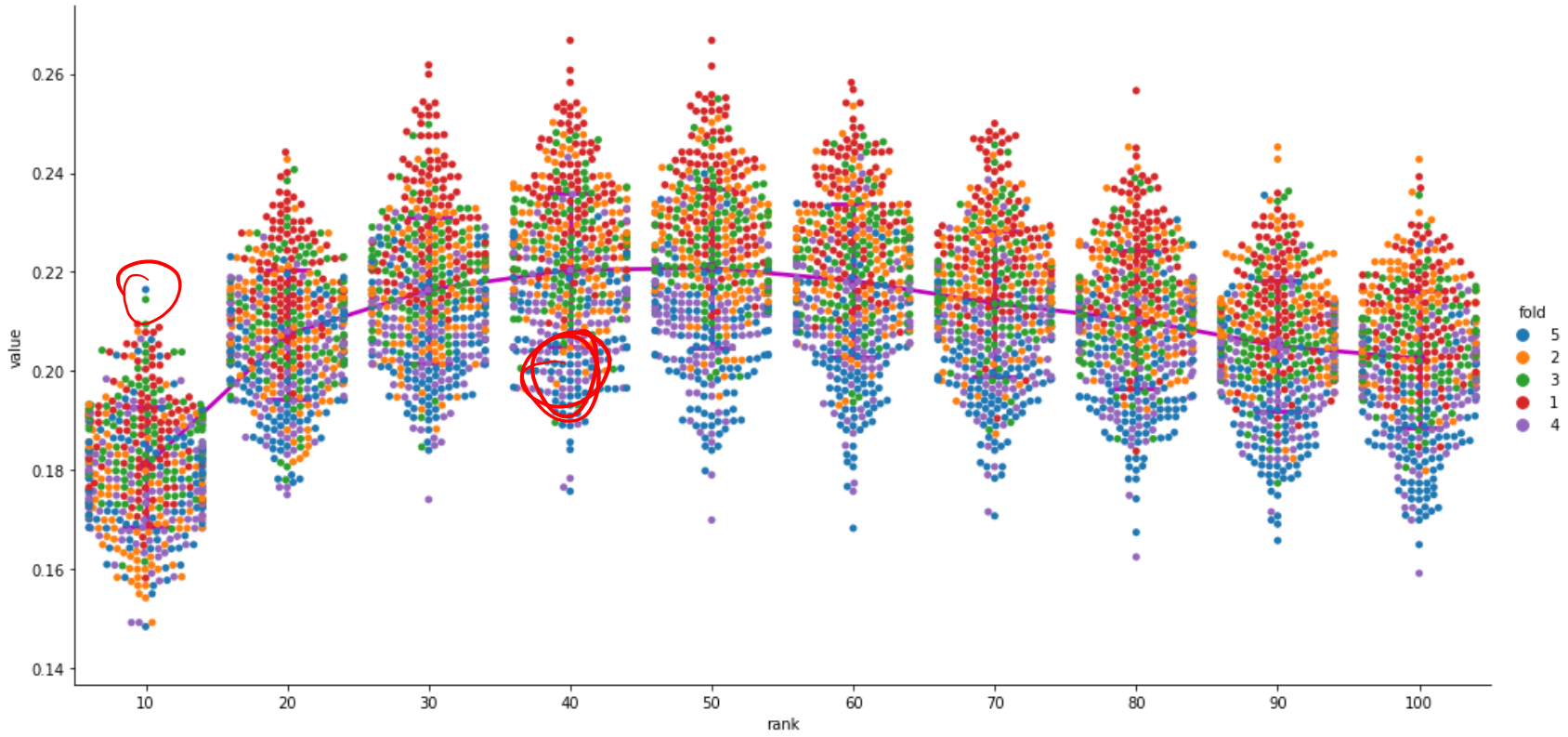


Quick (potentially biased) CV



Effects of data shuffling in CV

- repeat CV many times with different initial random seed:
  - generate random seed
  - shuffle data and split into folds
  - run CV experiment and record metrics



Temporal splits

Data split strategy	Definition of training and test instances	Local timeline	Global timeline	Data leakage
Random-split-by-ratio	Randomly sample a percentage of user-item interactions as test instances; the remaining are training instances.	No	No	Yes
Random-split-by-user	Randomly sample a percentage of users, and take all their interactions as test instances; the remaining instances from other users are training instances.	No	No	Yes
Leave-one-out-split	Take each user's last interaction as a test instance; all remaining interactions are training instances.	Yes	No	Yes
Split-by-timepoint	All interactions after a time point are test instances; interactions before this time point are training instances.	No	Yes	No

Table source: Ji, Yitong, Aixin Sun, Jie Zhang, and Chenliang Li. "A critical study on data leakage in recommender system offline evaluation." *arXiv preprint arXiv:2010.11060* (2020).

How often do algorithms recommend items from future?

Timepoint violation metric:

$$\frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\sum_{i\in R_u}\mathbb{I}(t_i^f > t_u^l)$$

t_i^f - the first time that item i appears in training set
t_u^l - the time of the last interaction from user u in the training set
R_u – ordered set of recommended items for user u

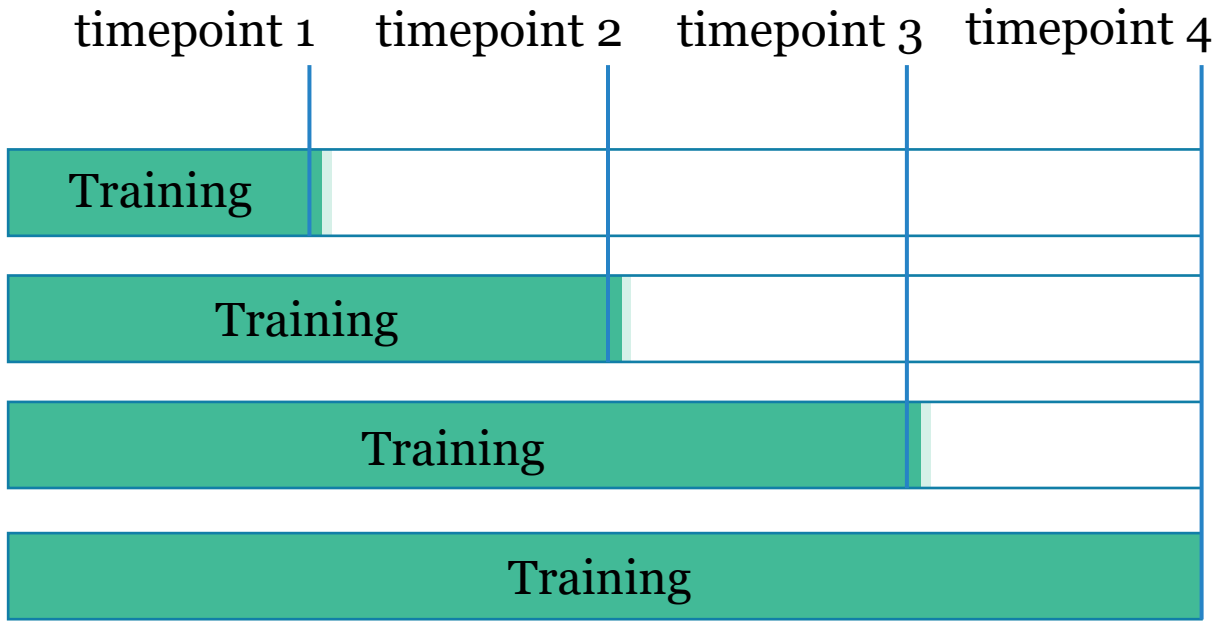
Table 6. The Number of Invalid Recommendations of Each User under the Four Splitting Methods (Averaged over Different Algorithms)

Dataset	#invalid			
	RO_RS	**RO_LS**	**TO_RS**	**TO_LS**
ML-1M	0.054 ± 0.004	0.049 ± 0.004	0.059 ± 0.004	0.054 ± 0.004
Yelp	0.067 ± 0.005	0.053 ± 0.005	0.078 ± 0.006	0.062 ± 0.005
Netflix	0.036 ± 0.000	0.034 ± 0.000	0.074 ± 0.000	0.056 ± 0.000
AMZ_Movie	0.169 ± 0.014	0.247 ± 0.023	0.195 ± 0.015	0.277 ± 0.023
AMZ_Video	0.283 ± 0.022	0.491 ± 0.038	0.414 ± 0.039	0.433 ± 0.040
AMZ_Toys	0.167 ± 0.014	0.353 ± 0.026	0.233 ± 0.020	0.403 ± 0.024
AMZ_Elec	0.079 ± 0.006	0.228 ± 0.019	0.269 ± 0.018	0.361 ± 0.021

RO_RS - Random Ordering Ratio-based Splitting
RO_LS - Random Ordering Leave-one-out Splitting
TO_RS - Temporal Ordering Ratio-based Splitting
TO_LS - Temporal Ordering Leave-one-out Splitting

Table source: Zhao, W. X., Lin, Z., Feng, Z., Wang, P., & Wen, J.-R. (2022). A Revisiting Study of Appropriate Offline Evaluation for Top-N Recommendation Algorithms . *ACM Transactions on Information Systems*, *41*(2). https://doi.org/10.1145/3545796

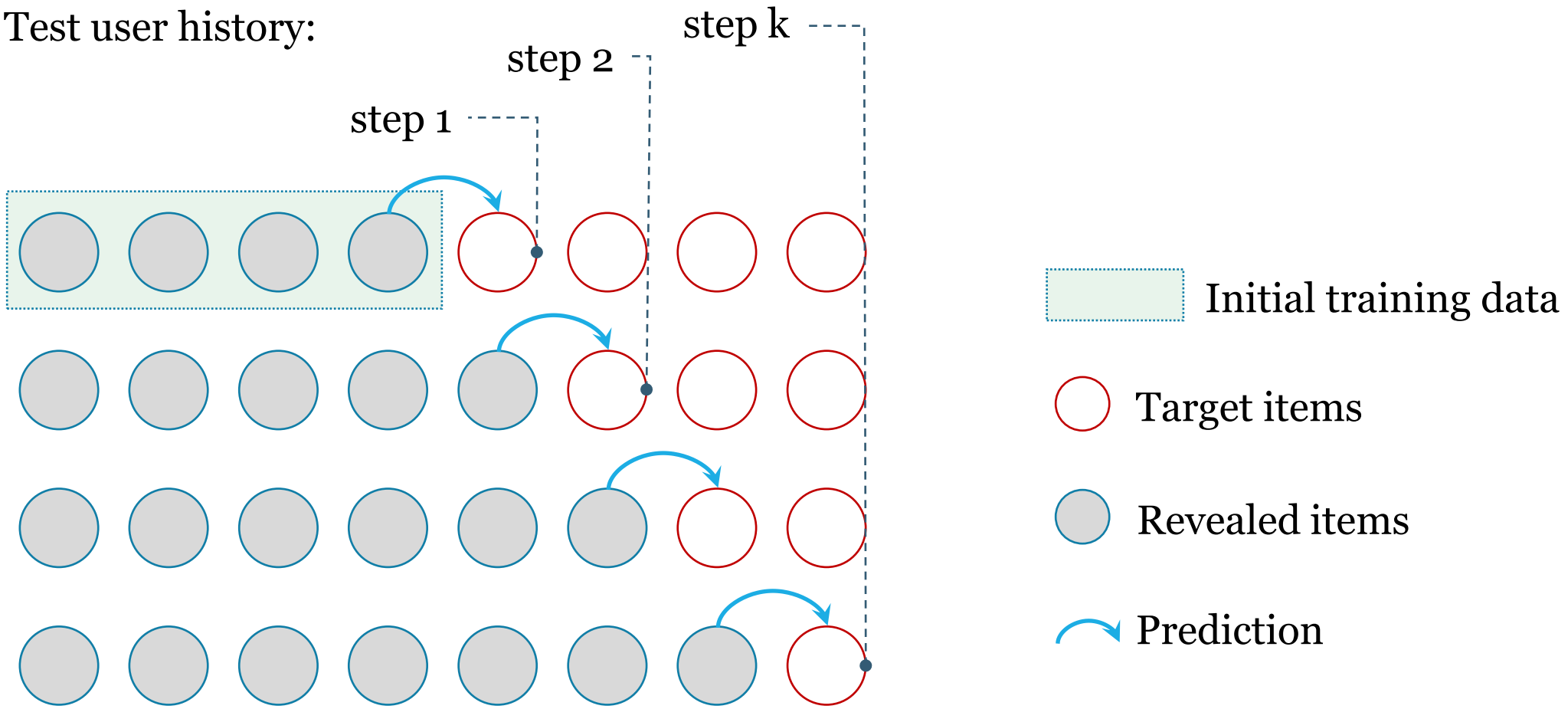
Temporal splits - successive evaluation



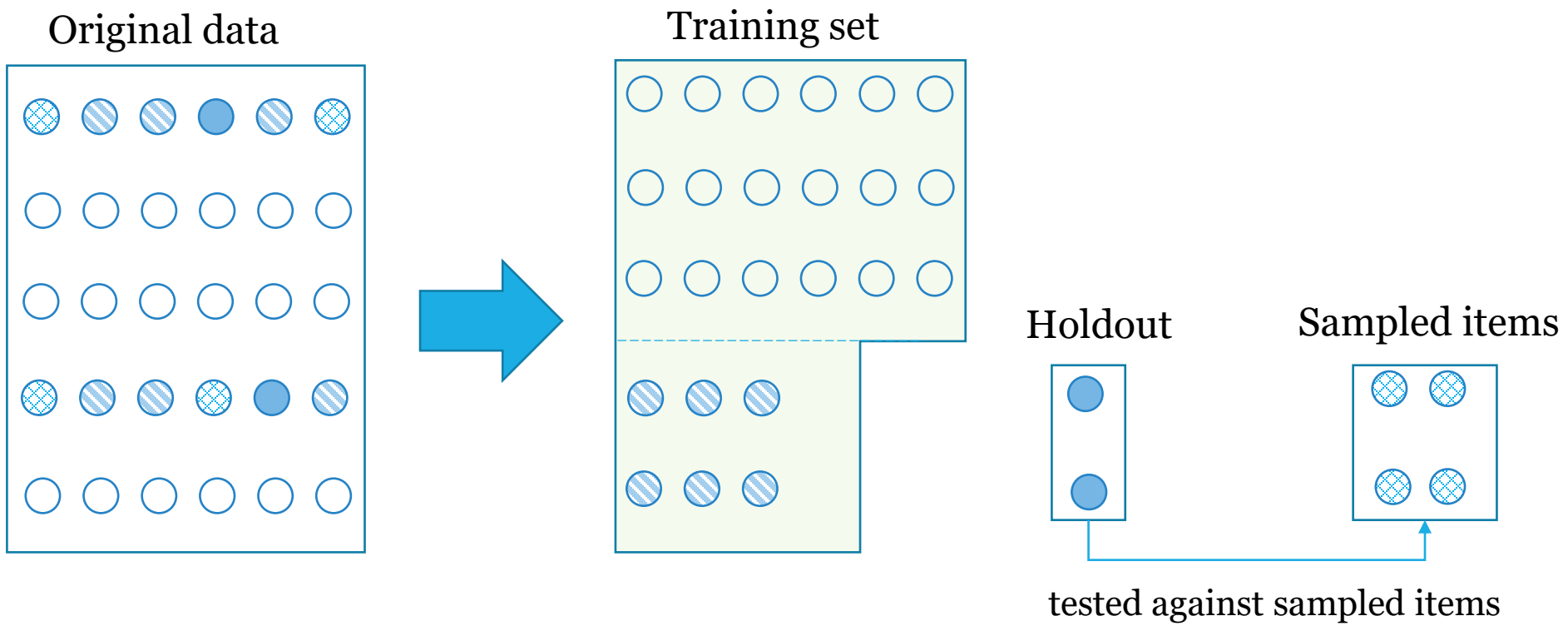
Timepoints at:

- next item
  - potentially leaky
  - OK with folding-in
- next session
- next day/week/month/year

Successive next-item revealing scheme



Target items sampling



Unrated items, selected randomly

Target items sampling issues

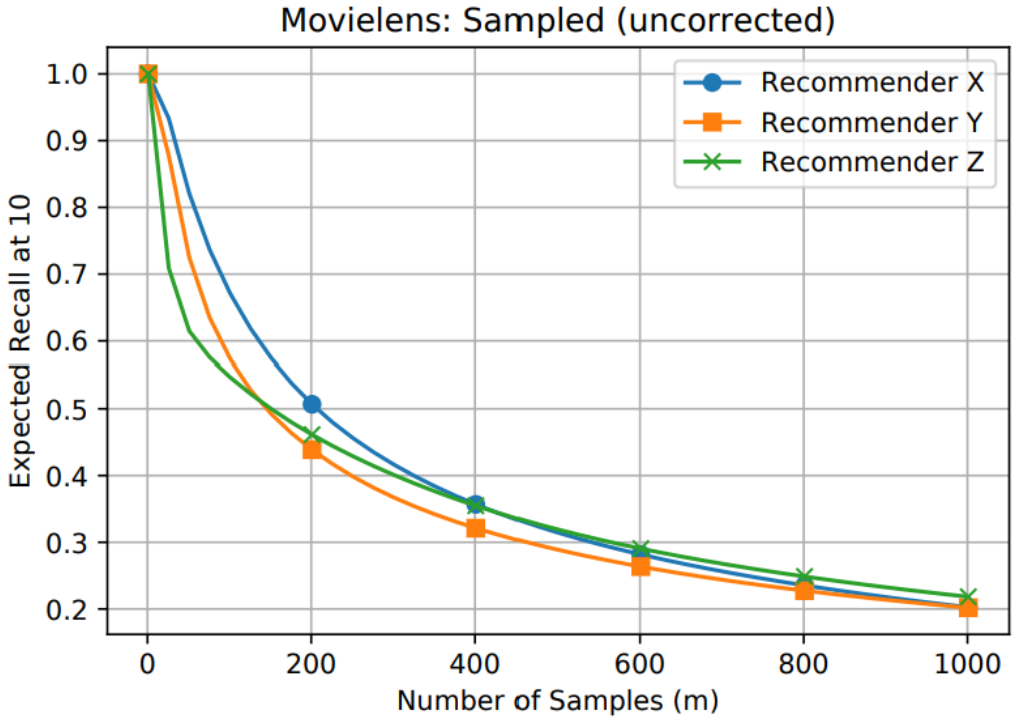


Image source: Krichene, Walid, and Steffen Rendle. "On sampled metrics for item recommendation." In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1748-1757. 2020.

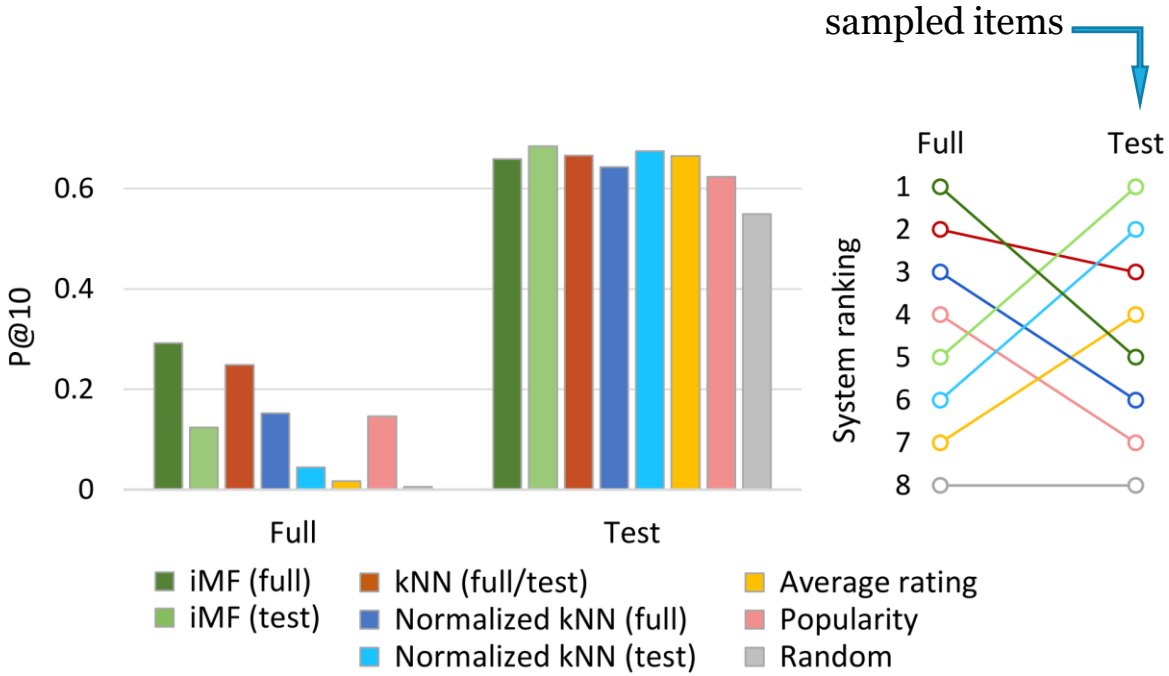


Image source: Cañamares, Rocío, and Pablo Castells. "On target item sampling in offline recommender system evaluation." In *Fourteenth ACM Conference on Recommender Systems*, pp. 259-268. 2020.

Methodological chaos

Table 1. Overview of data splitting strategies reported in the literature, as well as the dataset(s) those papers use.

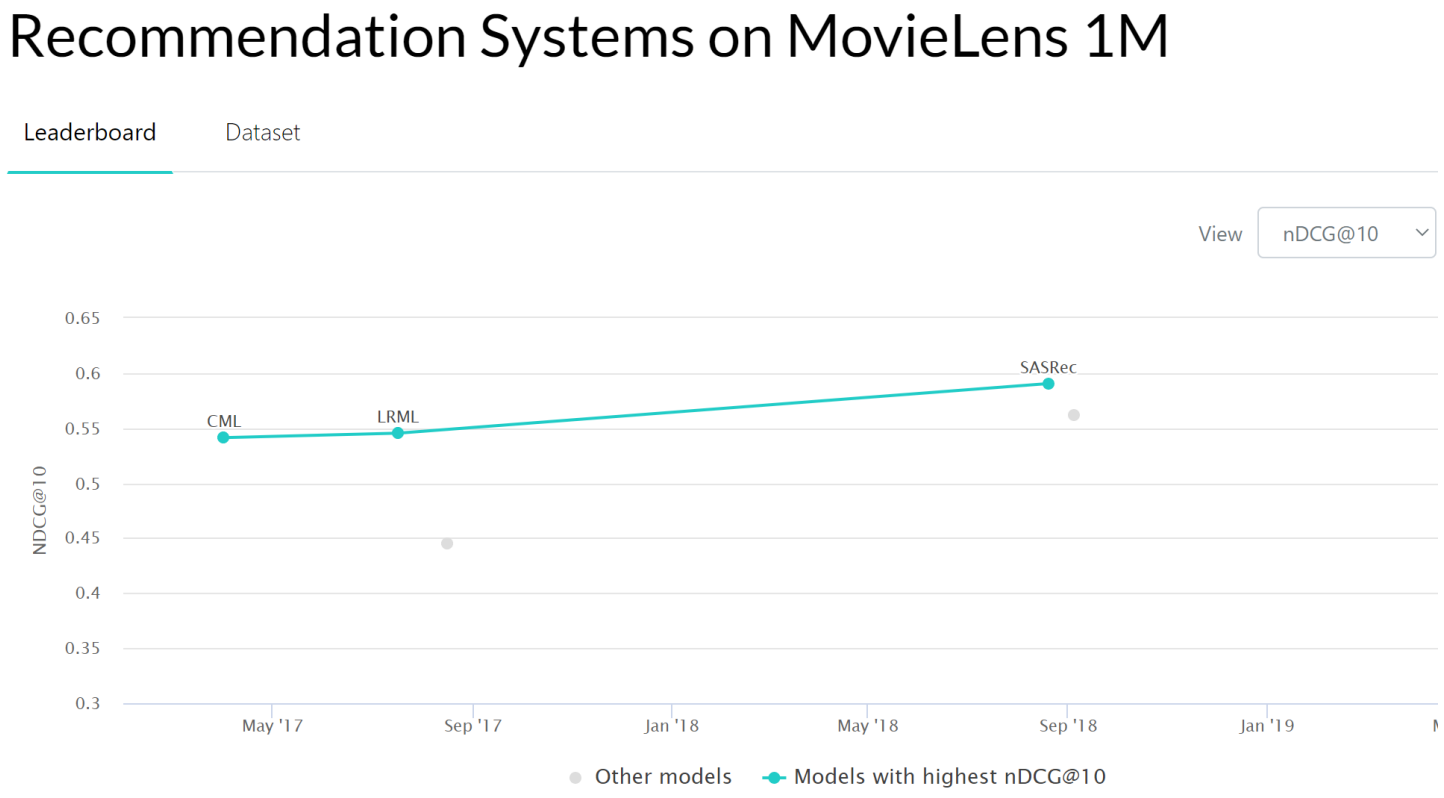
Model	Leave One Last		Temporal Split		Random Split	User Split	Used Datasets
	Item	Basket/Session	User-based	Global			
BPR [13] (2009)	×	×	×	×	√	×	N
FPMC [14] (2010)	×	√	×	×	×	×	-
NeuMF [4] (2017)	√	×	×	×	×	×	M1, P
VAECF [9] ((2018))	×	×	√	×	×	√	M2, N
Triple2vec [22] (2018)	×	√	×	×	×	×	I, D
SARRec [6] (2018)	√	×	×	×	×	×	A, M1
CTRec [1] (2019)	√	√	√	×	×	×	T, A
SVAE [17] (2019)	×	×	√	×	×	√	M1, N
BERT4Rec [20](2019)	√	×	×	×	×	×	A, M1, M2
NGCF [24] (2019)	×	×	×	×	√	×	A, G, Y
VBCAR [11] (2019)	×	×	×	√	×	×	I
KGAT [23] (2019)	×	×	√	×	×	×	A, Y
Set2Set [5] (2019)	×	×	√	×	×	×	T, D
DCRL [25] (2019)	×	×	×	√	×	×	M2, G
TiSASRec [8] (2020)	√	×	×	×	×	×	M1, A
JSR [26] (2020)	√	×	×	×	×	×	M2, A
HashGNN [21] (2020)	×	×	×	×	√	×	M2, A

M1: Movielens-1M, M2: Movielens-20M, **T**: Tafeng, D: Dunnhumby
G: Gowalla, I: Instacart N: Netflix, A: Amazon, Y: Yelp, P: Pinterest

Table from: Meng, Zaiqiao, Richard McCreadie, Craig Macdonald, and Iadh Ounis. "Exploring data splitting strategies for the evaluation of recommendation models." In *Fourteenth ACM conference on recommender systems*, pp. 681-686. 2020.

Public leaderboards are (mostly) inconsistent

Example: Papers With Code
https://paperswithcode.com/sota/collaborative-filtering-on-movielens-1m



CML paper:

4.2 Evaluation Methodology

We divide each user's ratings into training/validation/test sets in a 60%/20%/20% split. Users who have less than 5 ratings are only included in the training set. The predicted ranking is evaluated on recall rates for Top-K recommendations, which are widely-used performance metrics for implicit feedback [46, 47].

SASRec paper:

We followed the same preprocessing procedure from [1], [19], [21]. For all datasets, we treat the presence of a review or rating as implicit feedback (i.e., the user interacted with the item) and use timestamps to determine the sequence order of actions. We discard users and items with fewer than 5 related actions. For partitioning, we split the historical sequence $\mathcal{S}^u$ for each user u into three parts: (1) the most recent action $\mathcal{S}^u_{|\mathcal{S}^u|}$ for testing, (2) the second most recent action $\mathcal{S}^u_{|\mathcal{S}^u|-1}$ for validation, and (3) all remaining actions for training. Note that during testing, the input sequences contain training actions and the validation action.

To avoid heavy computation on all user-item pairs, we followed the strategy in [14], [48]. For each user u, we randomly sample 100 negative items, and rank these items with the ground-truth item. Based on the rankings of these 101 items, Hit@10 and NDCG@10 can be evaluated.

Significance Testing

- t-test
- Paired t-test
- Wilcoxon test
- …

Table 2
Paper count and percentage for each category

Category	# Papers	% of Papers
Used specified test	26	23
Used confidence interval	8	7
Used unspecified test	12	11
No significance test	65	59

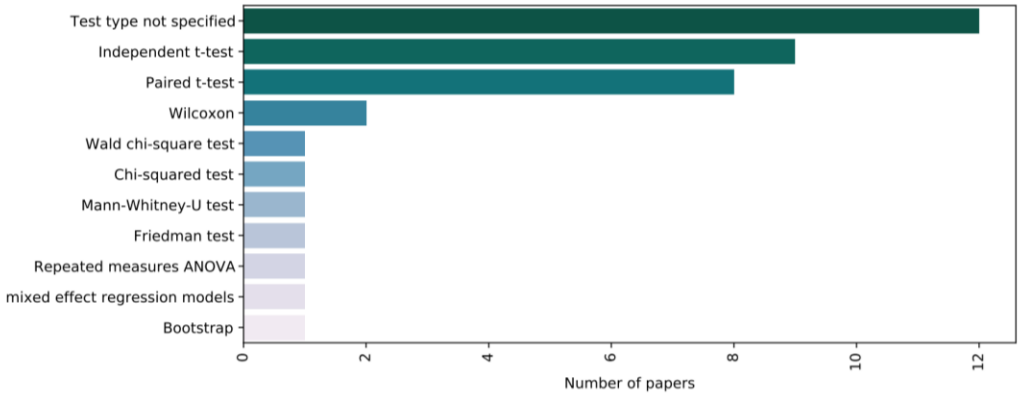


Figure 1: Distribution of selected papers coded as – used significance test.

Source: Ihemelandu, Ngozi, and Michael D. Ekstrand. "Statistical Inference: The Missing Piece of RecSys Experiment Reliability Discourse." In *Proceedings of the Perspectives on the Evaluation of Recommender Systems Workshop, CEUR Workshop, vol. 2955, 2021.*

“The Field Needs Better, More Unified and Harder Evaluation”

Reading:

- Dacrema, Maurizio Ferrari, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. "**A troubling analysis of reproducibility and progress in recommender systems research.**" *ACM Transactions on Information Systems (TOIS)* 39, no. 2 (2021): 1-49.

- Zhang, Shuai, Lina Yao, Aixin Sun, and Yi Tay. "**Deep learning based recommender system: A survey and new perspectives.**" *ACM Computing Surveys (CSUR)* 52, no. 1 (2019): 1-38.

Netflix prize story

Contest: Given a database of movies rated by users, beat Netflix's recsys by at least 10%

October 2, 2006 - June 26, 2009

Award: $1,000,000

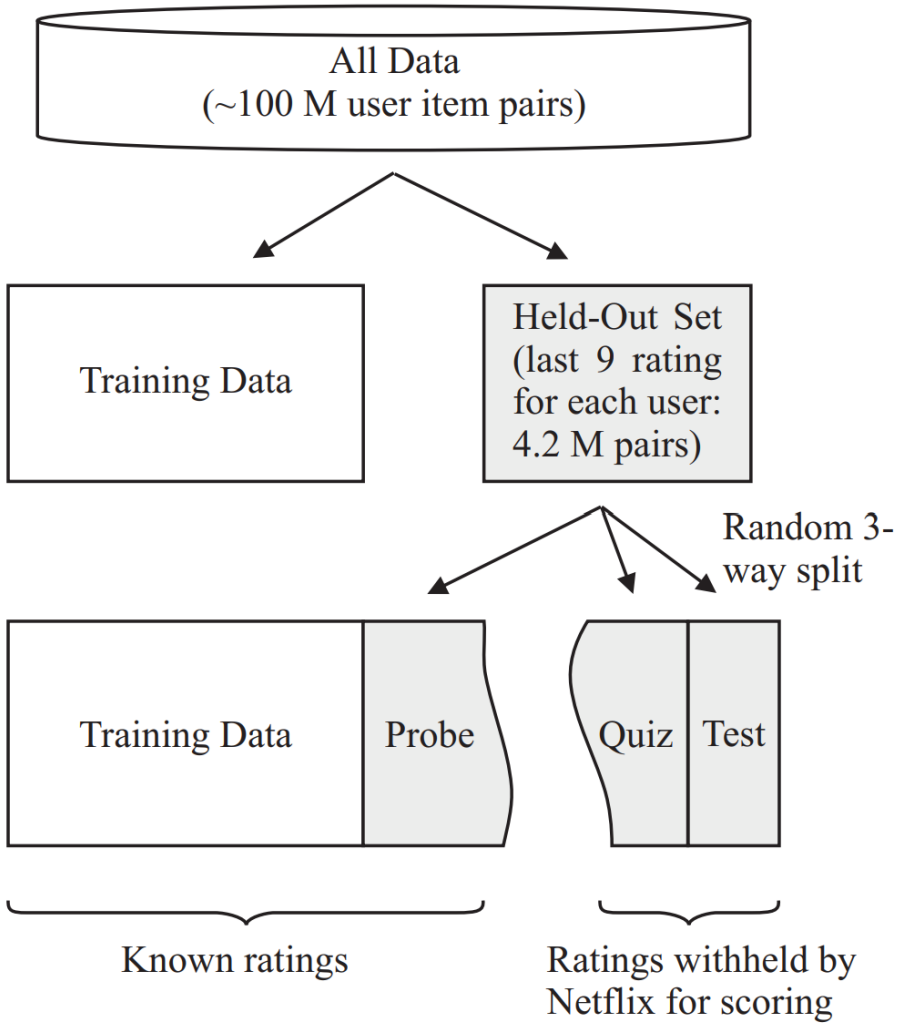
Key to success: ensemble of models.
Actual solution was never implemented!
https://www.techdirt.com/blog/innovation/articles/20120409/03412518422/why-netflix-never-implemented-algorithm-that-won-netflix-1-million-challenge.shtml

But latent factors models based on **matrix factorization** gained popularity.

Issues:

- treated as a pure matrix completion problem
- evaluation metric**:** RMSE

Netflix Prize Evaluation Scheme



Let's find issues and leaks in methodology:

Image source: Takács, Gábor, István Pilászy, Bottyán Németh, and Domonkos Tikk. "Scalable collaborative filtering approaches for large recommender systems." *The Journal of Machine Learning Research* 10 (2009): 623-656.

Metrics

Metrics

Error-based
RMSE, MAE

Relevance based
precision, recall
F1-score
HR (hit rate)

Ranking-based
nDGC (normalized discounted cumulative gain)
MAP (mean average precision)
MRR (mean reciprocal rank)
ATOP (area under the TOPK-curve)
AUC (area under the ROC-curve)

Other aspects:

- Coverage
- Novelty
- Serendipity
- Diversity
- Trust
- Utility

Penalizing error vs rewarding accuracy

- Error metrics usually penalize errors, whereas accuracy metrics reward correct answers
- Asymmetry: not useful to have low error on irrelevant items, if error is high on relevant items.

Examples:

- RMSE vs. HR, Recall
- AUC vs. NDCG, MAP

Error minimization vs recommendations accuracy

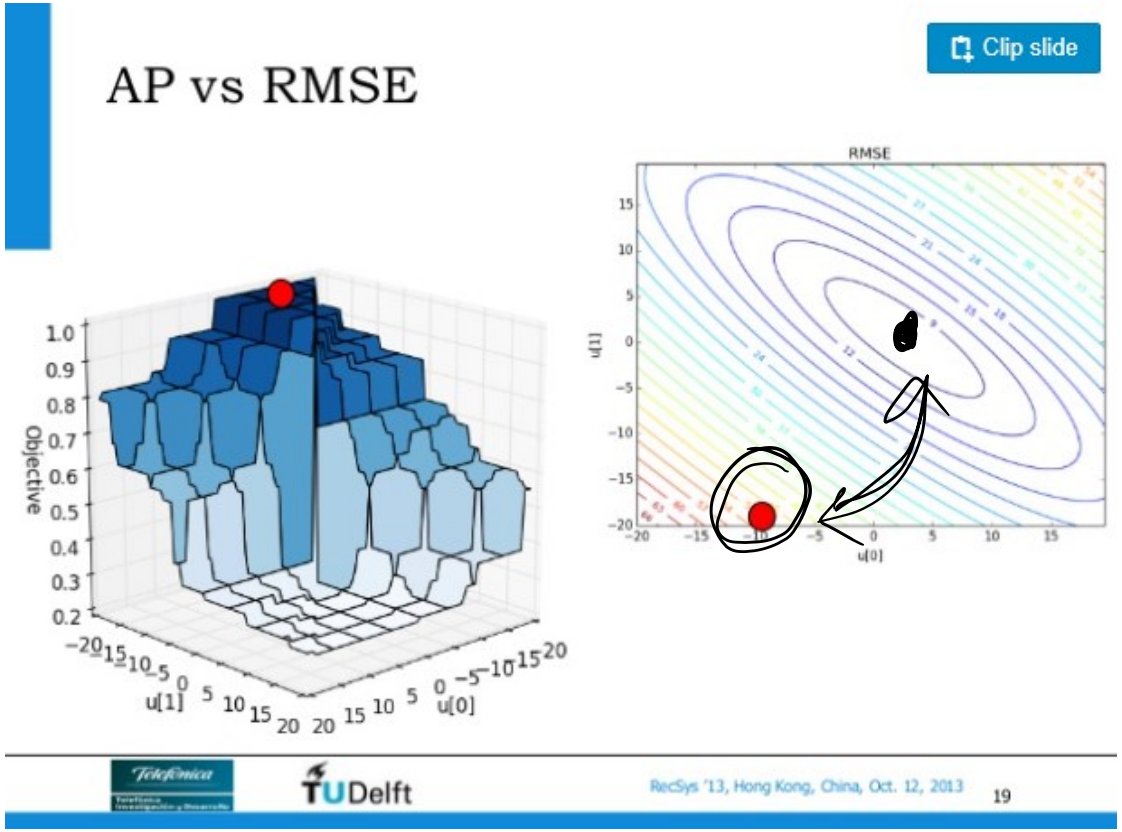


Figure credit: http://www.slideshare.net/kerveros99/learning-to-rank-for-recommender-system-tutorial-acm-recsys-2013

Rank-correlation metrics

Examples:

- Spearman (tie-corrected)
- Kendall Tau

Only useful if

- the recommendations list is short or
- ranking in the end of the list is equally important as in the beginning.

Metrics calculation chaos

Library	HitRate@20	MAP@20	MRR@20	NDCG@20	Precision@20	Recall@20	RocAuc
RePlay	0.475	0.039	0.186	0.093	0.058	0.096	0.283
DL RS Evaluation	1.15	0.039	0.186	0.08	0.058	0.096	0.283
DaisyRec	0.475	0.023*	0.275	0.093	0.058	0.096	fail
Beta RecSys	—	0.032	—	0.093	0.058	0.096	0.687
RecBole	0.475	0.039	0.186	0.093	0.058	0.096	0.687
Elliot	0.475	0.073	0.186	0.09	0.058	0.096	0.688
OpenRec	—	—	—	0.463	0.058	0.096	0.705
MS Recommenders	—	0.032	—	0.093	0.058	0.096	0.687
NeuRec	0.475*	0.003	0.186	0.093	0.058	0.096	—
rs_metrics	0.475	0.032	0.186	0.093	0.058	0.096	—

Table 1: Metrics calculated using different libraries.

Table source: Tamm, Y. M., Damdinov, R., & Vasilev, A. (2021). Quality metrics in recommender systems: Do We calculate metrics consistently? *RecSys 2021 - 15th ACM Conference on Recommender Systems*, 708–713.

Top-n recommendations task

Expected output: a ranked list of n most relevant items.

$$\text{toprec}(i, n) := \arg\max_{j}^{n} r_{ij}$$

r_{ij} - is the predicted relevance score between user i and item j

Example:

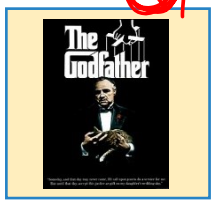
r_{ij}	0.73	0.41	0.95
item			

top-2 recommendations

leftsorted list

Metrics calculation example

User



holdout

known user preferences

recommendations

Algorithm:

$$\text{Precision} = \frac{1}{\#(\text{test users})} \sum_{\substack{\text{test} \\ \text{users}}} \frac{\#\,(\text{recommended items} \cap \text{holdout items})}{\#\,(\text{recommended items})}$$

$$\text{Recall} = \frac{1}{\#(\text{test users})} \sum_{\substack{\text{test} \\ \text{users}}} \frac{\#\,(\text{recommended items} \cap \text{holdout items})}{\#\,(\text{holdout items})}$$

Denoted as *metric@n*, where n = #recommended items, e.g. Recall@n

Metrics calculation example

Assume #{holdout items} = const = 1, then Recall ≡ HitRate:

HitRate:

$$\mathrm{HR@}n = \frac{1}{\#(\text{test users})} \sum_{\substack{\text{test} \\ \text{users}}} \text{hit}$$

$$\text{hit} = \begin{cases} 1 & \text{if any holdout item is among } n \text{ recommended items,} \\ 0 & \text{otherwise.} \end{cases}$$

Mean Reciprocal Rank:

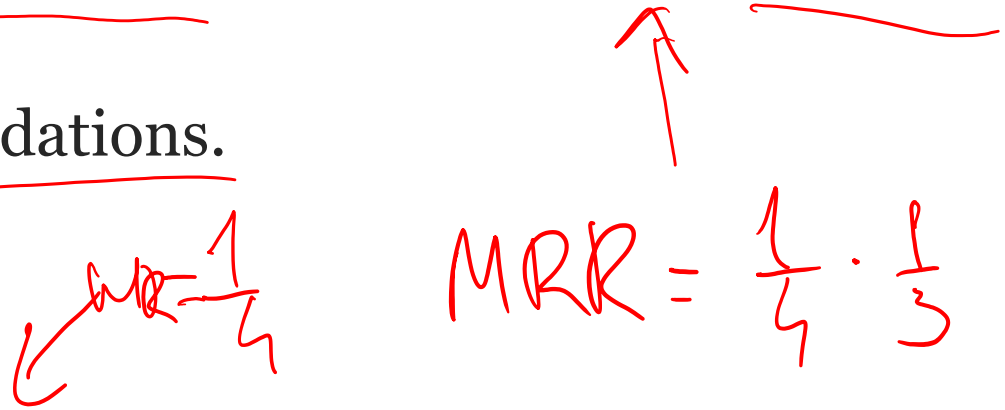
$$\mathrm{MRR@}n = \frac{1}{\#(\text{test users})} \sum_{\substack{\text{test} \\ \text{users}}} \text{hit} \cdot \frac{1}{\text{hit rank}}$$

hit rank = earliest position of a holdout item in the left-ordered list of n recommendations

Task

Description:

- There're 4 test users.
- Each user is presented with a list of top-4 personal recommendations.
- One user got a single correct recommendation and it's located at the 3rd position in the list.
- Other users got no correct recommendations.



Questions:

- What are HR and MRR in that case?
- How will HR change for top-5 recommendations (assuming no new correct guesses)?
- How will HR change for top-2 recommendations?

HR and MRR in general case

Assume #{holdout items} ≠ 1

Helpful notation:

U_{test} – set of test users

H_u – set of items in holdout for user u

R_u – ordered set of recommended items for user u

$\text{rank}(u, i)$ – position of item i in R_u

For top-n recommendations, $|R_u| =$

$$\text{HR@}n = \frac{1}{|U_{\text{test}}|} \sum_{u \in U_{\text{test}}} \mathbb{I}(R_u \cap H_u \neq \emptyset)$$ ← hit

$$\text{MRR@}n = \frac{1}{|U_{\text{test}}|} \sum_{u \in U_{\text{test}}} \frac{\mathbb{I}(R_u \cap H_u \neq \emptyset)}{\min(\text{rank}(u, i))}$$

Recall

Helpful notation:
U_{test} – set of test users
H_u – set of items in holdout for user u
R_u – ordered set of recommended items for user u

$$\text{Recall@}n = \frac{1}{|U_{\text{test}}|} \sum_{u \in U_{\text{test}}} \frac{|R_u \cap H_u|}{|H_u|}$$

Ranking metrics: nDCG

$$DCG@n = \frac{1}{|U_{\text{test}}|} \sum_{u \in U_{\text{test}}} \sum_{i \in R_u} \frac{2^{r_{ui}} - 1}{\log_2(\text{rank}(u,i) + 1)}$$

r_{ui} - true rating of a recommended item, assuming $r_{ui} = 0$ if item is not rated

$$nDCG@n = \frac{DCG@n}{iDCG@n}$$

$iDCG = DCG$ with "ideal" (based on true ratings) ranking of items

Alternative (linear) weighting: $\frac{r_{ui}}{\log_2(\text{rank}(u,i)+1)}$

Ranking metrics: MAP

Mean Average Precision:

$$\text{MAP@}n = \frac{1}{|U_{\text{test}}|} \sum_{u \in U_{\text{test}}} \frac{1}{|R_u \cap H_u|} \sum_{i \in R_u \cap H_u} \text{P}_u\text{@rank}(u, i)$$

P_u - precision for user u with partial R_u

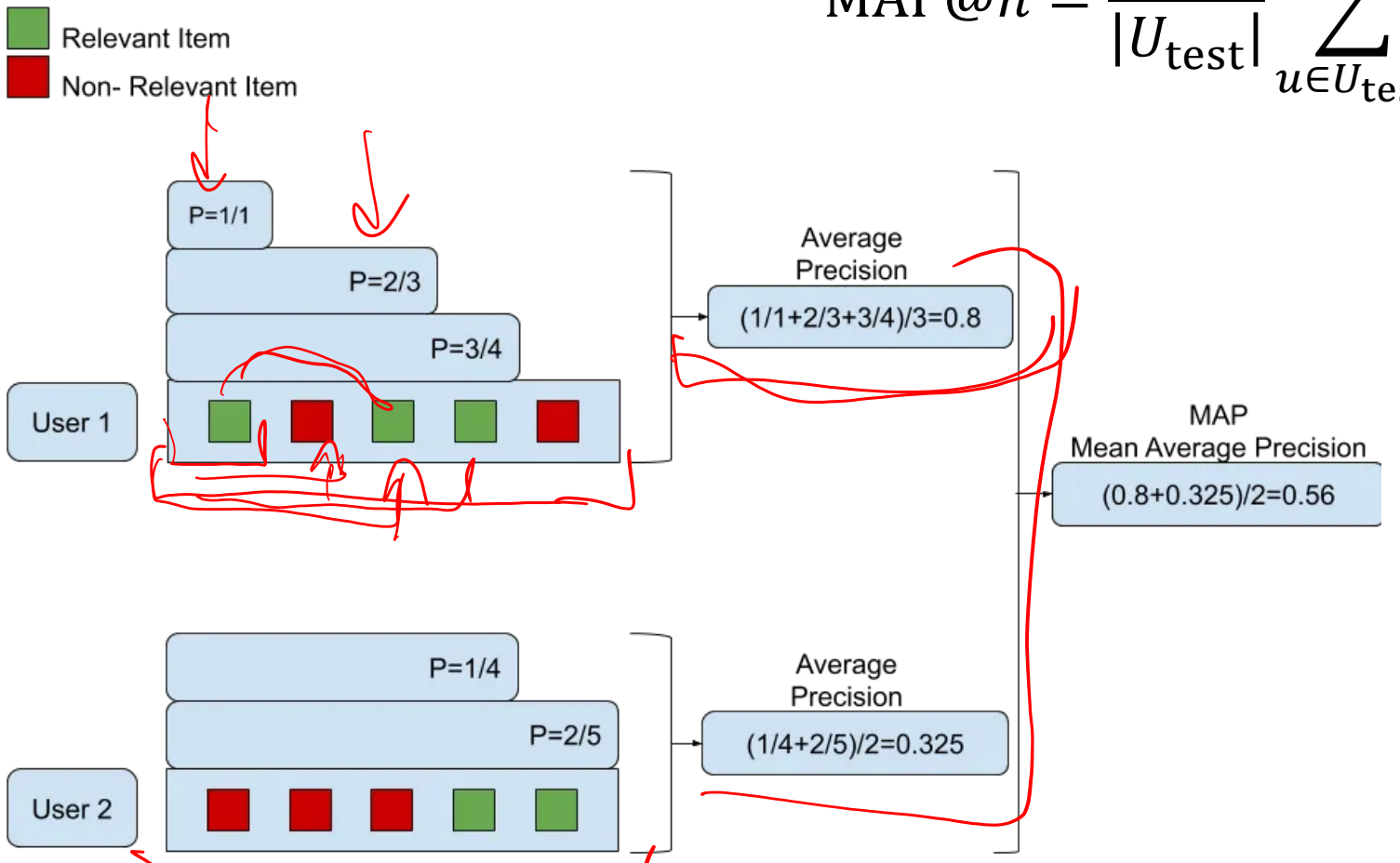


Image credit: https://medium.com/swlh/rank-aware-recsys-evaluation-metrics-5191bba16832

Useful metric: Coverage

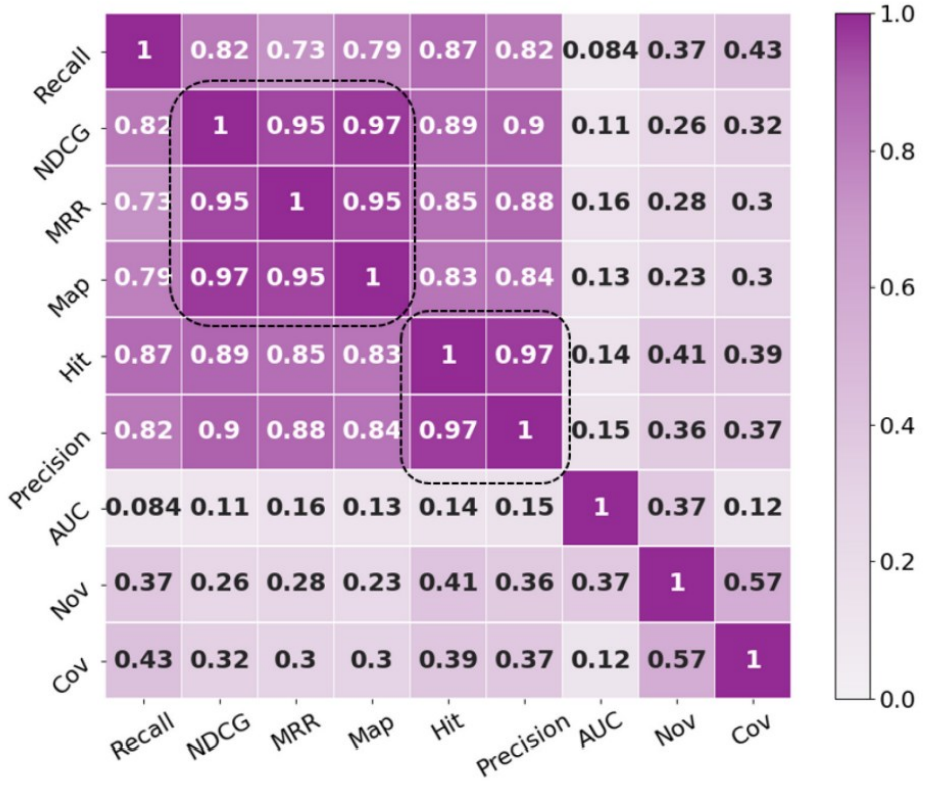
- fraction of unique recommendations in entire item catalog:

$$\text{COV@}n = \frac{\left|\bigcup_{u \in U_{\text{test}}} R_u\right|}{|I|}$$

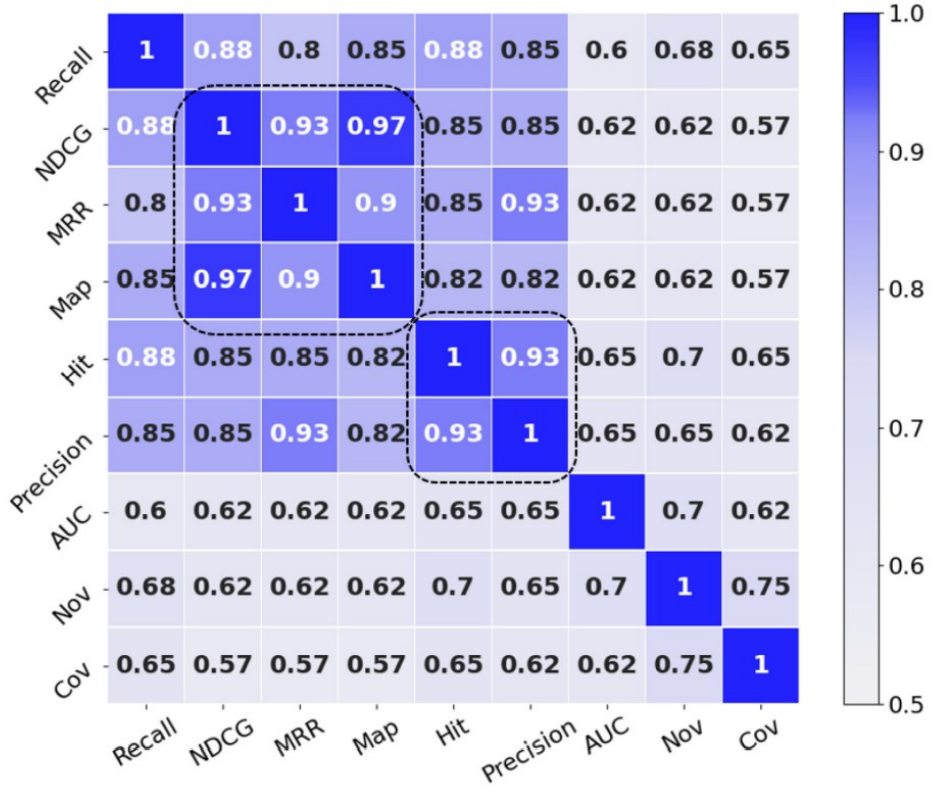
I – set of all items

- simple metric to measure diversity of recommendations
- enables quick analysis of recommendations adequacy
  - lower values – potentially too obvious recommendations
  - high values – potentially too random recommendations

Metric correlations



Spearman Rank Correlation



Overlap Ratio

Image source: Zhao, W. X., Lin, Z., Feng, Z., Wang, P., & Wen, J.-R. (2022). A Revisiting Study of Appropriate Offline Evaluation for Top- N Recommendation Algorithms . *ACM Transactions on Information Systems*, *41*(2). https://doi.org/10.1145/3545796

Task

Consider top-2 recommender for 10 users from 20-items (2 items per user).

What's better:
1) Correctly recommend 2 items to each of 5 users?
or
2) 1 item to each of 10 users?

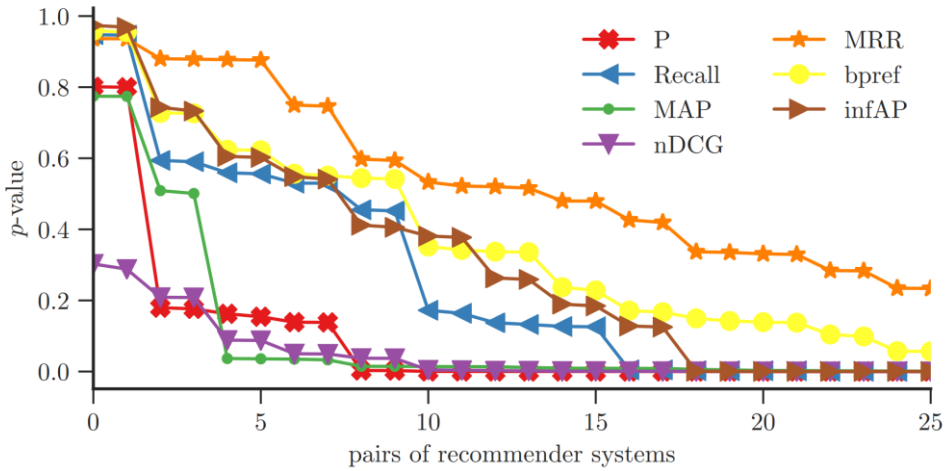
Why?

Use standard definition of precision and recall.

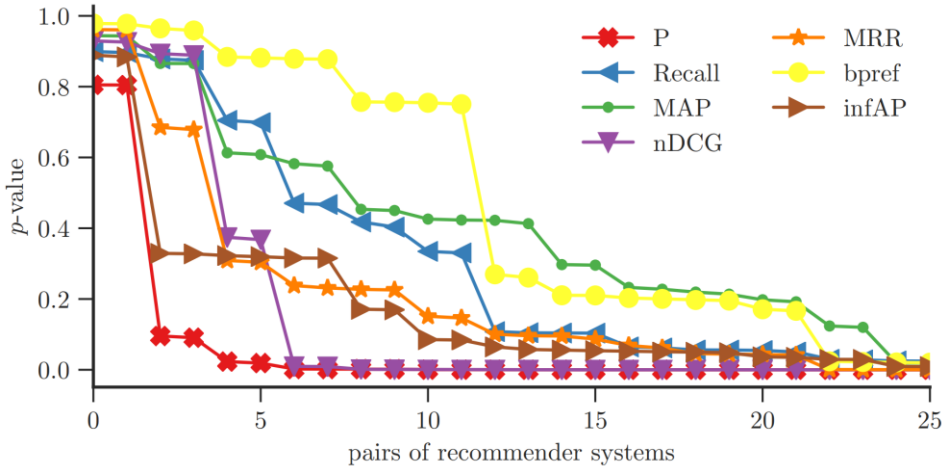
A note on discriminative power of metrics

Protocol:

- compare two recommendation techniques
- variation in the metric values is expected to indicate a statistically significant difference
- use statistical tests to measure the significance (via *p*-value, the lower the better)



a: MovieLens 1M.



c: BeerAdvocate.

Image credit: "*On the robustness and discriminative power of information retrieval metrics for top-N recommendation*" Daniel Valcarce et al. In Proceedings of the 12th ACM Conference on Recommender Systems (2018) https://dl.acm.org/citation.cfm?doid=3240323.3240347