From Variability to Stability: Advancing RecSys Benchmarking Practices

Valeriy Shevchenko
Skoltech
Moscow, Russian Federation

Nikita Belousov
Skoltech
Moscow, Russian Federation

Alexey Vasilev
Sber, AI Lab
Moscow, Russian Federation

Vladimir Zholobov
Skoltech
Moscow, Russian Federation

Artyom Sosedka
Sber, Sber AI
Moscow, Russian Federation

Natalia Semenova
Sber, Sber AI
Moscow, Russian Federation

Anna Volodkevich
Sber, AI Lab
Moscow, Russian Federation

Andrey Savchenko
Sber, AI Lab
Moscow, Russian Federation

Alexey Zaytsev
Skoltech, BIMSA
Moscow, Russian Federation

ABSTRACT

In the rapidly evolving domain of Recommender Systems (RecSys), new algorithms frequently claim state-of-the-art performance based on evaluations over a limited set of arbitrarily selected datasets. However, this approach may fail to holistically reflect their effectiveness due to the significant impact of dataset characteristics on algorithm performance. Addressing this deficiency, this paper introduces a novel benchmarking methodology to facilitate a fair and robust comparison of RecSys algorithms, thereby advancing evaluation practices. By utilizing a diverse set of 30 open datasets, including two introduced in this work, and evaluating 11 collaborative filtering algorithms across 9 metrics, we critically examine the influence of dataset characteristics on algorithm performance. We further investigate the feasibility of aggregating outcomes from multiple datasets into a unified ranking. Through rigorous experimental analysis, we validate the reliability of our methodology under the variability of datasets, offering a benchmarking strategy that balances quality and computational demands. This methodology enables a fair yet effective means of evaluating RecSys algorithms, providing valuable guidance for future research endeavors.

ACM Reference Format:
Valeriy Shevchenko, Nikita Belousov, Alexey Vasilev, Vladimir Zholobov, Artyom Sosedka, Natalia Semenova, Anna Volodkevich, Andrey Savchenko, and Alexey Zaytsev. 2024. From Variability to Stability: Advancing RecSys Benchmarking Practices. In *Proceedings of The 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24).* ACM, New York, NY, USA, 17 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Recommender systems have become the backbone of personalizing user experiences across diverse online platforms. Suggesting movies on streaming services, proposing products for purchase, and curating personalized news feeds [1], RecSys is one of the core machine learning technologies widely used in applications. Their impact drives ongoing development in both academia and industry, resulting in the introduction of numerous RecSys algorithms each year [2].

With this ongoing expansion, there is a growing need for tools that enable reproducible evaluation, allowing researchers to assess new methods alongside well-established baselines [3–5]. While several frameworks [6–8] excel in conducting a rigorous evaluation of RecSys algorithms on a specific dataset, selecting the best-performing models across multiple problems remains challenging. Results vary significantly based on the considered dataset, and what works well in one context may perform poorly in another [9]. This variability often results in inconsistent conclusions from evaluation studies, highlighting the importance of comparing algorithms across datasets with various data characteristics. On the other hand, extensive evaluation with dozens of datasets uses high amounts of computational resources — and harms both the environment and opportunities for small research labs. With researchers seeking universally effective algorithms across different recommendation tasks, businesses question algorithms' performance on datasets that reflect their specific industry domain or characteristics, trying to shorten time-to-production for RecSys.

However, in contrast with other machine learning subdomains like time series classification [10] and NLP [11], the field of RecSys lacks an accepted performance aggregation method across multiple datasets. Furthermore, there is limited research dedicated to comparing and contrasting different recommendation datasets, understanding their impact on the performance of RecSys algorithms, and identifying datasets with similar characteristics.

To deal with these problems, we develop a comprehensive benchmark methodology that can reliably rank RecSys methods based on their performance across various problems using offline evaluation, overcoming the limitations of current practices. Our approach confidently determines whether a specific top-1 model can excel universally or within particular domains defined by dataset characteristics. While providing reliable results, we use only a small number of datasets, enabling robust and efficient comparison.

Our contributions include:

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

- A benchmarking methodology tailored to the RecSys domain, featuring a clearly defined evaluation protocol and hyperparameter tuning with the swift integration of new algorithms [1].
- Utilization of 30 public datasets for benchmarking. Among them, there are two new large-scale open-source datasets from distinct RecSys domains (music and e-commerce), accessible for download [2].
- Comparative analysis of various metrics aggregation methods and their robustness stress tests to identify the most suitable approach for RecSys multi-dataset benchmarking.
- Investigation into the relationship between specific dataset characteristics and recommendation quality and identification of dataset clusters with similar characteristics.
- Efficient comparison procedure that uses only 6 datasets but provides similar ranking due to reasonable selection of benchmarking datasets based on clustering.
- Identification of the top-performing algorithms from a pool of 11 frequently used approaches based on principled metrics aggregation across multiple scenarios.

2 RELATED WORK

RecSys evaluation. Recommender systems continue to be a dynamic research area. Traditional techniques like Neighborhood-based models [12, 13] and Matrix Factorization [14] remain reliable baselines. However, incorporating Deep Neural Networks has notably advanced RecSys, significantly enriching the domain [15]. This variety leads to the development of open-source libraries and tools to address diverse application needs. Among the noteworthy ones, DeepRec [16], Implicit [17], LightFM [18], NeuRec [19], RecBole [20], RecPack [21] and Replay [22] offer realizations of popular recommendation algorithms.

Offline evaluation remains essential in RecSys research as it provides a reliable and cost-effective approach to assess algorithm performance. It is particularly suitable for researchers who are developing new models. As a part of offline evaluation, the variety in the field leads to the need for rigorous and reproducible evaluation methodologies. Notable studies such as Elliot [6], Recbole [7], and DaisyRec [8] introduced comprehensive evaluation frameworks in both reproducing and benchmarking recommendation models. These frameworks offer a rich array of options for pre-filtering, data splitting, evaluation metrics, and hyperparameter tuning across a broad spectrum of popular recommender models. Notably, Elliot uniquely provides statistical tests for robustly analyzing the final results, adding a layer to the evaluation process.

RecSys datasets. Dozens of public datasets from diverse domains are available for constructing and evaluating recommender systems. The research [23] shows that most studies utilize, on average, 2.9 datasets, with dataset selection and preprocessing affecting evaluation outcomes significantly. Different data filtering techniques can change data characteristics, leading to varied performance rankings. Deldjoo et al. [24, 25] investigated how data properties impact recommendation accuracy, fairness, and vulnerability to shilling attacks, highlighting the importance of data understanding in enhancing system performance. The paper [9] emphasized the crucial role of dataset diversity in RecSys algorithm evaluations, showing that dataset choice significantly affects evaluation conclusions. These findings collectively underscore the need for considering dataset variability in future research to enhance the reliability of recommender system evaluations.

Aggregating methods. When introducing a novel machine learning approach, it is important to rigorously compare its performance against existing methods across a comprehensive set of relevant tasks to determine its standing relative to the current state-of-the-art. However, drawing conclusions about the superior algorithm based on the outcomes from multi-dataset benchmarks can be challenging.

Various techniques have been developed to yield concise summaries to address this challenge. One basic method involves mean aggregation, assuming uniformity across task metrics [26]. However, this can lead to biased evaluations when metrics vary significantly [27]. The Dolan-Moré performance profiles, initially developed for optimization algorithm benchmarks [28], have gained traction for evaluating machine learning algorithm efficacy across diverse problems [29]. Unlike mean aggregation, Dolan-Moré curves consider the distribution of performance values, offering insight into how frequently and significantly one algorithm excels. Similarly, the Critical Difference (CD) diagram [30] is frequently used to compare algorithms across multiple tasks. This method of presenting results has become broadly accepted [31]. The CD diagram provides both groupwise and pairwise comparisons. Groupwise comparisons are achieved by ordering all methods based on the mean rank of relative performance on each task. Pairwise comparisons are based on a critical difference in this mean rank or, in newer version [32], the Wilcoxon signed-rank test [33] with multiple test corrections.

VOTE'N'RANK [11] is another framework proposed for ranking systems in multitask benchmarks rooted in the principles of social choice theory. The framework employs scoring and majority-relation-based rules, such as Plurality, Dowdall, Borda, Copeland, and Minimax, to ensure a more comprehensive evaluation.

Benchmarking is a fundamental practice in machine learning, crucial for measuring progress through datasets, metrics, and aggregation methodologies to evaluate system performance. These benchmarks are crucial for comparing new algorithms with established ones to identify the most effective models for practical use.

Performance benchmarks are essential across various fields. For example, the ILSVRC (ImageNet Large Scale Visual Recognition Challenge) [34] consider object classification and detection with extensive image datasets and unique metrics for each task. In natural language processing (NLP), GLUE [35] and its derivatives [36] benchmark models across diverse tasks, ranking them based on mean score values. One example in the AutoML domain is AMLB [37], which emphasizes multitask evaluation via mean ranking.

The research [38] offers an in-depth and reproducible evaluation of ten collaborative filtering algorithms, employing a Borda count ranking method to aggregate accuracy results from various datasets and metrics. The study emphasizes that, although this method provides valuable insights, it necessitates careful interpretation

[1] To guarantee the reproducibility of our results, all code and datasets employed in experiments are available in the GitHub repositoryhttps://anonymous.4open.science/r/recsys-evaluation-of-dozens-datasets-0635.
[2] The link will be available after peer-review.

due to biases favouring algorithms that perform well in correlated metrics.

To the best of our knowledge, BARS [39] is the most advanced benchmarking initiative focused on RecSys. Although BARS establishes an open benchmark with standardized evaluation protocols, it presents certain limitations. For instance, it restricts itself to only three datasets dedicated to the singular challenge of top-N recommendation, maintaining discrete leaderboards for each dataset. Such an approach, devoid of a multi-dataset scoring mechanism, inhibits discerning truly adaptive and universal models — covering this gap might offer significant insights for researchers.

3 METHODOLOGY

Our aim is to present a robust and efficient benchmarking methodology tailored for the RecSys domain. We align our experimental setup with online evaluation, replicating real-time recommendation scenarios while ensuring the reproducibility of our benchmarking results.

To achieve our goal, we collect a diverse set of open-source datasets and establish a robust pipeline that incorporates predefined procedural steps. Additionally, we integrate 11 RecSys algorithms from various open-source libraries and repositories. This pipeline serves a dual purpose: it streamlines the evaluation process and enhances the comparability of results across different algorithms and datasets. The pipeline scheme is shown in Figure 1.

3.1 Datasets and Preprocessing

In our benchmarking process, we use 30 public datasets, each with timestamps, spanning seven diverse domains. These datasets cover many business areas, including e-commerce, social networks, and entertainment. Alongside the utilization of the 28 established public datasets, we introduce two new ones, namely Zvuk and SMM, with their details provided in the Appendix. This diversity is summarized in Table 1.

Implicit feedback-based recommendation systems are increasingly prevalent, primarily due to the frequent absence of explicit rating information in many applications. Therefore, datasets that initially include item ratings are usually transformed into binary signals, an approach we have also implemented in our evaluation. [8, 38]. We have introduced a dataset-specific threshold parameter, denoted as τ, to filter out interactions falling below this threshold. Such interactions are considered negative feedback and are thus removed from the datasets. For more on determining the optimal τ value for individual datasets, refer to Appendix A.

In their initial state, the datasets exhibit a highly sparse nature, characterized by a substantial proportion of users interacting with a limited number of items, often fewer than five. As part of the evaluation process, preprocessing steps are applied to filter out inactive users and items. Most researchers either adopt a 5- or 10-filter/core preprocessing [3, 8]. F-filter and F-core filtering techniques differ. The former simultaneously filters items and users in a single pass, while the latter employs iterative filtering until all users and items have a minimum of F interactions. We adopt the 5-filter [3] methodology, prioritizing item filtering before user filtering. Thus, each user has a minimum of 5 interactions, but some items might have fewer.

[3]For the newly introduced datasets, Zvuk and SMM, we have implemented a 50-filter due to their extensive size.

Domain	#	Datasets
Movies and clips	9	Movielense (1, 10, 20M) [40], Netflix [41], Douban movies [42], Amazon TV [43], Kuairec (full/small) [44], Rekko [45]
Food and beverage	5	BeerAdvocate [46], RateBeer [47], Food [48], Amazon FineFoods [43], Tafeng [49]
Social networks (SN)	4	Yelp review [50], Epinions [51], RedditHyperlinks [52], DianPing [53]
Books	3	MTS library [54], Douban books [42], GoodReads[55, 56]
Location-based SN	3	Gowalla [57], Brightkite [57], FourSquare [58]
Music	3 + **1**	Amazon CDs [43], Amazon Musical Instruments [43], Douban music [42], **Zvuk [new]**
E-commerce	1 + **1**	Retailrocket [59], **SMM [new]**

Table 1: Dataset distribution by domains, # is the number of datasets in a domain. The newly introduced *Zvuk* and *SMM* expand the most data-scarce domains

3.2 Recommendation Models

Current recommendation frameworks enable the streamlined integration of widely-used baseline models and recently proposed models. We have leveraged existing implementations of well-known algorithms and developed an evaluation pipeline. This pipeline encompasses dataset filtering, data splitting, metrics computation, and hyperparameter optimization. The frameworks we have used include Implicit [17], LightFM [18], RecBole [20], and Replay [22].

Reflecting on recent relevant research in benchmarking [8, 38], we have selected the following categories of algorithms for our analysis:

- Non-personalized baseline: Random and Popularity-based recommendations (**Random** and **MostPop**).
- Neighborhood-based model: **ItemKNN** [13].
- Matrix factorization models: **LightFM** [18], **ALS** [60], and **BPR** [61].
- Linear models: **SLIM** [62] and **EASE** [63].
- Neural models: **MultiVAE** [64], **LightGCN** [65], and **LightGCL** [66]

For an in-depth overview of the compared algorithms and their corresponding hyperparameters, please refer to Appendix D. While our selection covers many recent approaches, one can add new algorithms from various sources, thereby expanding the scope and capability of the benchmark.

3.3 Evaluation Settings

Data splitting. A guiding principle for splitting data into training and test subsets is to resemble the deployment conditions closely [67]. In the top-N recommendation paradigm, the primary

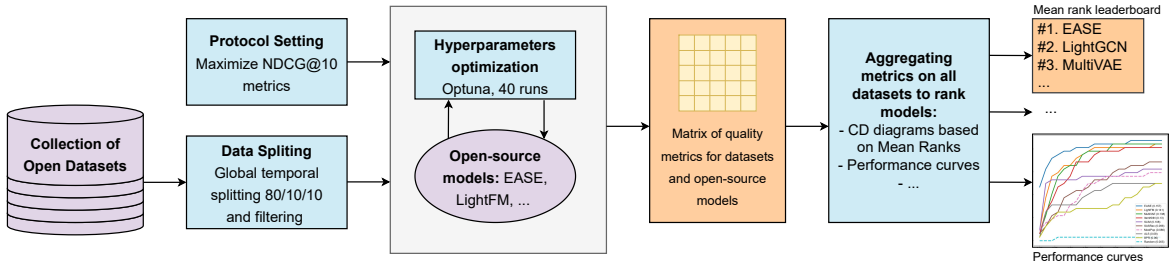


Figure 1: The proposed benchmarking for the ranking of algorithms. Our main innovations are the curated list of datasets that enable the option of comparison of pairs of models and aggregation strategies that provide principled ranking of approaches w.r.t. various criteria.

challenge is to infer user preferences from past interactions to predict future ones. Given this, the training data should chronologically precede the test data, acting as the "history" followed by the "future" at a designated time. This approach helps in mitigating the risk of data leakage [68]. Therefore, we adopt the global temporal splitting strategy with an 80/10/10 training, validation, and test set ratios following [68–70]. After splitting, we exclude cold-start users and items with no record in the training set.

Negative Sampling. In the context of Recommender Systems evaluation, negative sampling involves prediction and evaluation for only a limited set of non-relevant items and known relevant items instead of full item list scoring. These non-relevant items are chosen from a candidate item pool. Although sampling strategies like the Uniform Sampler have been used to avoid biases in evaluating RecSys algorithms and to boost computational efficiency [8, 10, 71], studies have questioned their reliability [1]. Consequently, our evaluation involves testing on all unobserved items.

Evaluation Metrics. The precise interpretation of popular quality metrics in the field lacks a consensus, and it is often observed that the more complex a metric is, the greater the scope for varying interpretations [72]. Therefore, offering a detailed evaluation protocol for reproducibility and clarity in the assessment is crucial. In light of this, our approach is meticulous: we precisely define each metric and accurately compute them within our established pipeline.

To evaluate the performance of our models, we employ a spectrum of standard quality metrics, such as *Precision@k*, *Recall@k*, *nDCG@k*, *MAP@k*, *HitRate@k*, and *MRR@k*. Our evaluation scope extends further, incorporating beyond-accuracy objective metrics that provide a more comprehensive view of model effectiveness. These include *Coverage@k*, *Diversity@k*, and *Novelty@k* [73]. For details on computation for each metric, please refer to Appendix D.

Hyperparameter Tuning. Hyperparameter optimization is crucial for achieving optimal performance of machine learning algorithms and ensuring reliable benchmarking. The paper [74] highlights that most RecSys baselines can attain between 90 – 95% of their maximum performance within the initial 40 iterations when using Bayesian optimization. Leveraging this insight, we utilize the Optuna framework [75] and apply the Tree of Parzen Estimators (TPE) algorithm for hyperparameter tuning. The hyperparameter search space can be found in Appendix B. In alignment with prior

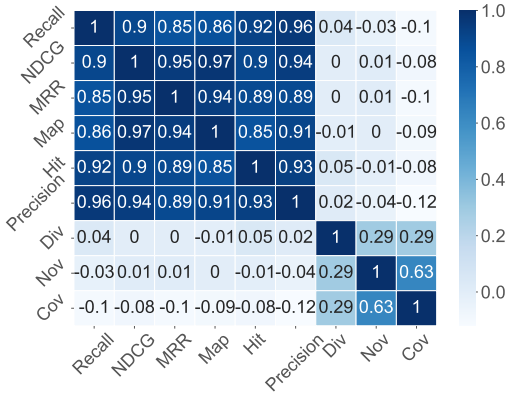


Figure 2: Spearman correlation between metrics for $k = 10$.

research [3, 23, 76], we conduct hyper-parameter optimization with respect to *nDCG@10* for each baseline on each dataset.

After determining the optimal hyperparameters, we execute a final training on the consolidated training and validation sets. This procedure ensures that all available interactions up to the test timestamp are incorporated, including the most recent ones.

3.4 Metrics Comparison Approach

In our benchmarking process, we utilize a matrix containing acquired metrics from various datasets and apply multiple comparison approaches to analyze these data.

Once evaluation metrics are collected, we should define a method to rank algorithms using performance score. Our pipeline uses well-established methods to aggregate performance to a single rank score over multiple datasets. These aggregations are adopted from general Machine learning practice and reused for our problem of RecSys methods ranking.

The list of aggregators includes arithmetic, geometric, and harmonic mean aggregations of a quality metric, CD diagrams [30] emphasizing mean ranks; Dolan-Moré (performance) curves [28] featuring AUC values, and algorithms developed in the principles of the social choice theory, specifically the Copeland, and Min-Max rules, proposed for aggregation of results over various NLP tasks [11].

4 EXPERIMENTS AND RESULTS

4.1 Metrics

Our experiments take off with collecting performance metrics to evaluate recommendation algorithms. These metrics include User Preference Accuracy, Ranking Quality, and Beyond-Accuracy metrics. For example, for the Movielens-1m, they are in Table 6 for different values of k (5, 10, 20, 100). Remarkably, LightGCL achieves the best result for *nDCG*@10.

Our analysis encompasses a broad scope as we evaluate a collection of 11 algorithms on 30 datasets. We proceed by analyzing the collected results. To accomplish this, we generate rankings using these metrics and calculate the Spearman correlation score between them This process is repeated for each dataset, and we compute the average ranking correlation scores. These results are consolidated into a correlation heatmap depicting relationships among all pairs of evaluation metrics. The resulting heatmap is shown in Figure 2, with darker colours indicating stronger correlations. Overall, the heatmap reveals that Accuracy and Ranking metrics located in the top left corner (Recall, nDCG, MRR, MAP, HitRate, Precision) exhibit high correlations with each other, surpassing 0.8, whereas the Beyond-Accuracy metrics at the bottom demonstrate weak correlations with the rest. This distinction can be attributed to the goal of these metrics: they do not describe recommendation quality.

Furthermore, one may conclude that nDCG has the highest correlation (≥ 0.9) with accuracy and ranking metrics. This reinforces the usage of *nDCG* in benchmarking, as well as during hyper-parameter optimization as an optimization objective. In the following sections, we utilize nDCG@10 in our experiments unless otherwise stated.

4.2 Comparative analysis of metrics aggregation methods

In this section, our primary focus is on exploring various methods to aggregate metrics derived from diverse datasets. We describe the considered aggregation approaches and then analyze the ranking of RecSys models using these methods.

Given the RecSys specifics [39, 77], we identify key requirements for a ranking method:

- **Ranking:** The benchmarking system should rank methods according to their performance.
- **Metric Value Consideration:** It should consider the metric values and their relative differences for specific problems, not just the relative positions.
- **Interpretability:** The results should be interpretable, providing clear insights into model comparisons.
- **Significance determination:** The system should explicitly define the significance of performance differences.
- **Agnosticism to adversarial manipulations:** The ranking should be robust to malicious influence.

4.2.1 Considered aggregation approaches. We consider the following ways to aggregate metrics:

Mean Ranks (MR). MR is used in Critical Difference diagrams and computes the average ranks of methods across all datasets.

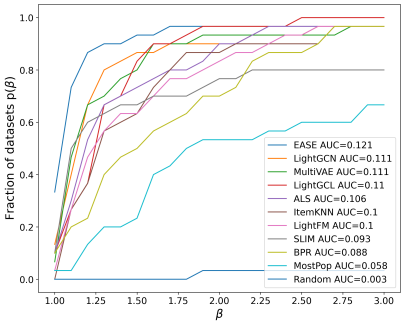


Figure 3: Performance profiles for the comparison of RecSys algorithms. The higher the curve, the better the performance of the algorithm. We also provide AUCs for each approach.

Mean Aggregations. These approaches utilize classic mathematical averages, such as *Mean aggregation (MA)*, *Geometric mean (Geom. mean)* and *Harmonic mean (HM)*, calculating across datasets for each model as a single model score.

Dolan-Moré Area Under Curve (DM-AUC). This relates to the Dolan-Moré performance profiles defined for $\hat{\beta} \geq \beta \geq 1$ as

$$p_i(\beta) = \frac{1}{d}\left|\left\{t : \beta q_{ti} \geq \max_j q_{tj}\right\}\right|,$$

where i is the index of the curve that corresponds to a RecSys model, t is the problem index, d is the number of datasets, $\hat{\beta}$ is a hyperparameter limiting the values of β, q_{ti} is the metric value that corresponds to a RecSys model i and the problem t, and q_{tj} is the metric value that corresponds to a RecSys model j and the problem t. The DM curve for a specific β reports the number of problems for which the model performs no more than β times worse than the best model for that problem (e.g., $p_i(1)$ represents the share of problems where the i-th algorithm is the best).

In the case of aggregation, we take the area under the DM curve divided by the sum of all areas. In our experiments, we fixed $\hat{\beta} = 3$; below, we show that ranking remains stable across a wide range of β values.

Dolan-Moré leave-best-out (DM LBO). This approach ranks algorithms by performing the following steps:

(1) Calculate DM AUC (area under curve) scores;
(2) Choose the best method using DM AUC and remove it;
(3) The method that is removed is assigned a rank based on the iteration in which it was dropped;
(4) Repeat the previous steps using the remaining methods.

Social Choice Theory. The last two aggregating approaches considered, *Copeland* and *Minimax*, are majority-relation-based rules. A majority relation for two methods m_A, m_B ($m_A \succ m_B$) holds, if m_A has a higher metric value than m_B. *Copeland* method defines the aggregation score $u(m)$ as $u(m_A) = |L(m_A)| - |U(m_A)|$, where $L(m_A) = \{m | m_A \succ m\}, U(m_A) = \{m | m \succ m_A\}$. The *Minimax* uses a score $s(m_A, m_B)$, representing the number of datasets for which method m_A has a higher score than m_B. The aggregated score is given by $u(m_A) = -\max_B s(m_B, m_A)$.

Ranking position	DM AUC ↑	DM LBO ↓	Mean ranks ↓	MA ↓	Geom. mean ↓	Harm. mean ↓	Copeland ↑	Minimax ↑
1	EASE: 0.121	EASE: 1	EASE: 2.833	EASE: 0.069	EASE: 0.042	EASE: 0.023	EASE: 10.0	EASE: -0.0
2	LightGCN: 0.111	LightGCN: 2	MultiVAE: 4.067	LightGCL: 0.065	LightGCN: 0.038	LightGCN: 0.021	MultiVAE: 8.0	SLIM: -21.0
3	MultiVAE: 0.111	LightGCL: 3	LightGCN: 4.533	LightGCN: 0.064	LightGCL: 0.038	ALS: 0.02	LightGCN: 6.0	MultiVAE: -22.0
4	LightGCL: 0.11	MultiVAE: 4	SLIM: 5.167	MultiVAE: 0.061	MultiVAE: 0.038	LightGCL: 0.02	SLIM: 3.0	LightGCN: -22.0
5	ALS: 0.106	ALS: 5	ALS: 5.2	LightFM: 0.059	ALS: 0.035	MultiVAE: 0.02	ALS: 2.0	LightGCL: -23.0
6	ItemKNN: 0.1	ItemKNN: 6	LightGCL: 5.633	SLIM: 0.058	LightFM: 0.034	ItemKNN: 0.018	LightGCL: 0.0	ALS: -24.0
7	LightFM: 0.1	LightFM: 7	LightFM: 5.667	BPR: 0.057	ItemKNN: 0.033	LightFM: 0.017	LightFM: -1.0	BPR: -25.0
8	SLIM: 0.093	BPR: 8	ItemKNN: 6.1	ALS: 0.057	BPR: 0.03	BPR: 0.014	ItemKNN: -4.0	ItemKNN: -26.0
9	BPR: 0.088	SLIM: 9	BPR: 6.933	ItemKNN: 0.056	SLIM: 0.025	MostPop: 0.006	BPR: -6.0	LightFM: -26.0
10	MostPop: 0.058	MostPop: 10	MostPop: 9.067	MostPop: 0.041	MostPop: 0.017	SLIM: 0.003	MostPop: -8.0	MostPop: -29.0
11	Random: 0.003	Random: 11	Random: 10.8	Random: 0.007	Random: 0.001	Random: 0.0	Random: -10.0	Random: -30.0

Table 2: Rankings of RecSys methods according to different aggregation approaches with their respective scores. The leaderboard is based on nDCG@10 values.

	DM AUC	DM LBO	Mean ranks	MA	Geom. mean	Harm. mean	Copeland	Minimax
Pareto efficacy	+	+	+	+	+	+	+	+
Using small number of datasets	+	+	+	−	+	−	+	−
Using small number of methods	+	+	−	+	+	+	+	−
Adding a new similar method	+	+	−	+	+	+	−	−
Adding a new best method	−	+	+	+	+	+	+	−
Changing hyperparameters	−	−	NA	NA	NA	NA	NA	NA
Spearman's correlation, 5 datasets	0.799	0.785	0.825	0.717	0.834	0.756	0.816	0.525
Spearman's correlation, 10 datasets	0.895	0.887	0.912	0.825	0.899	0.885	0.907	0.767

Table 3: General results of rankings reliability. + stands for the availability of a feature, − stands for an absence, and NA stands for not applicable.

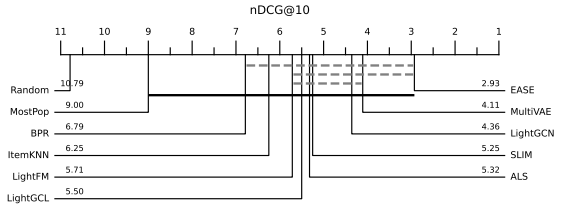


Figure 4: The Critical Difference diagram for the comparison of RecSys algorithms. The numbers represent the mean ranks of methods over all datasets. Thick horizontal lines represent a non-significance based on the Wilcoxon-Holmes test, while dashed horizontal lines represent non-significance according to the Bayesian Signed-Rank test.

4.2.2 Comparison of RecSys algorithms. One of our objectives is to present interpretable results that facilitate swift visual comparisons of the performance of RecSys algorithms across multiple datasets. We present these visual comparisons using Dolan-Moré Performance profiles (DM) and a Critical Difference (CD) diagram. The DM profiles are in Figure 3. Further, we use the presented DM AUC to rank the algorithms. The CD diagram can be found in Figure 4. In addition to the traditional CD diagram that includes the pairwise Wilcoxon test, we have introduced the Bayesian Signed-Rank test, indicated by dashed horizontal grey lines. We exclude the concept of ROPE from our analysis because it requires homogeneity among the set of metrics, which is not applicable in RecSys. This inhomogeneity also leads to the absence of statistical significance in the CD. While we use a large number of datasets, due to their diversity, the ranks of approaches change a lot.

However, our findings indicate that EASE emerges as the winner for both options. There is no distinct second-place algorithm, as areas under the performance profiles for LightGCN and MultiVAE are almost identical. Finally, all methods perform significantly better than a random approach and are mostly superior to the MostPop baseline.

Leaderboard for different aggregations. Different aggregation methods yield distinct rankings for the approaches considered. With the set of 30 datasets, the ranks presented in Table 2 consistently identify EASE as the top-performing approach. For the subsequent top positions, we have a pair of candidates for most aggregations: MultiVAE and LightGCN.

4.2.3 Comparison of reliability of aggregations. To ensure a reliable aggregation method for benchmarking, it should demonstrate stability under various perturbations, including adversarial ones. We conduct an analysis to assess the robustness of the presented aggregation methods.

First, we determine rankings based on 30 datasets across all methods, establishing them as our reference benchmarks. Next, we

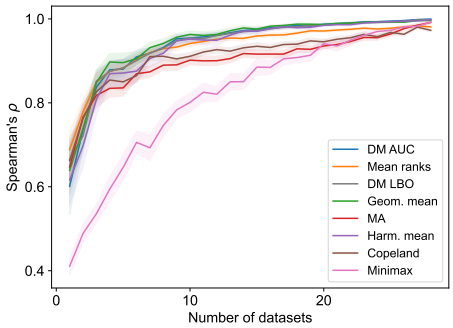


Figure 5: Stability of aggregations with respect to the number of used datasets.

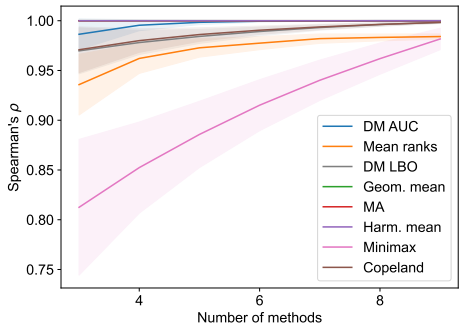


Figure 6: Stability of aggregations with respect to the number of used methods.

examine the sensitivity of these rankings to the following modifications of the input matrix of quality metrics:

(1) Inclusion or exclusion of a dataset.
(2) Introduction or removal of a RecSys algorithm.
(3) Incorporation or exclusion of a slightly superior/inferior method to a particular algorithm, exploring all possible permutations.
(4) Adjustments in the hyperparameters of an aggregation method.

Change of the set of used datasets. We measure the correlation between the final rankings and the references using the Spearman correlation coefficient ρ.

The results for the case of dropping datasets are presented in Figure 5. All aggregations exhibit a relatively stable behaviour, except for Minimax, which shows a low ρ after dropping 15 datasets. On the contrary, the best-performing method is Geom. mean, Harm. mean and DM LBO, with their average metric values being less influenced by specific datasets. Overall, different aggregation techniques tend to produce similar rankings if the number of datasets is large enough.

Furthermore, we explore the case when we use only five datasets to calculate ranks. We randomly sampled 100 pairs of subsets of size five and calculated Spearman's correlation between aggregations for each pair of sets of datasets. The results are in Table 3. Aggregations are less stable in this case. Moreover, *MA*, *Harm. mean* and *Minimax* methods have Spearman's correlation of less than 0.8. As in the case of dropping datasets, the outlier is *Minimax* model with low ρ. The *Mean ranks* and *Copeland* methods perform the best in this scenario, demonstrating equivalent ρ values.

Elimination of RecSys methods. In this scenario, we compute Spearman's correlation between the results for all methods and the results with the exclusion of some methods.

The results are presented in Figure 6. We observe that most aggregating methods exhibit relatively stable behaviour, except for *Minimax*. As the number of discarded methods increases, the likelihood of changing the best method also increases. This notably impacts the *Mean ranks*, *DM AUC* and *Copeland* methods.

Changing the hyperparameters of the aggregating method. In the paper, only the *DM AUC* and *DM LBO* aggregating methods have adjustable hyperparameters ($\hat{\beta}$ - the maximum value of the ratio of the best metric value and the one under consideration, the right boundary of the X-axis in the performance profiles). We calculate the Spearman correlation between the case when $\hat{\beta} = 3$ and the case when $\hat{\beta}$ can take any value. The results presented in Figure 10c demonstrate that $\hat{\beta}$ can influence the rankings, underscoring the significance of determining and fixing the optimal value. The ranking remains stable over a wide range of $\hat{\beta}$ values, with *DM LBO* showing more robustness compared to the pure *DM AUC*

Additional experiments. In the Appendix, section F.2, we explore the stability of a ranking provided by aggregations after the inclusion of a new method slightly superior or inferior to an existing one. There, all methods, except for *Mean ranks*, demonstrate stability under adversarial perturbations. Moreover, our analysis delves into specific intrinsic properties of a ranking system, such as Pareto efficiency when using a small number of datasets. An aggregation method is considered Pareto efficient if it outperforms another method for all metrics. The results of all tests are presented in Table 3.

4.3 Dataset characteristics

In addition to the performance benchmark, our study explores the connection between specific dataset characteristics and recommendation quality. We utilize user-item interaction matrix properties from [25]. These properties serve as problem characteristics and encompass various aspects, including the size, shape, and density of the dataset (*SpaceSize, Shape, Density*), as well as counts of users, items, and interactions (*Nu, Ni, Nr*). We also consider interaction frequencies per user and item (*Rpu, Rpi*), Gini coefficients that describe interaction distribution among users and items (*Giniu, Ginii*), and statistics related to popularity bias and long-tail items (*APB, StPB, SkPB, KuPB, LTavg, LTstd, LTsk, LTku*) [25]. The characteristics of the 30 datasets we selected for our analysis exhibit a wide range of variability, as shown in Figure 11.

To establish a connection between these data characteristics and our primary quality metric, *nDCG*@10, we employed three distinct measures: Pearson product-moment correlation, Spearman rank — an order correlation, and Mutual information — a nonlinear alternative. The obtained values are in Figure 7.

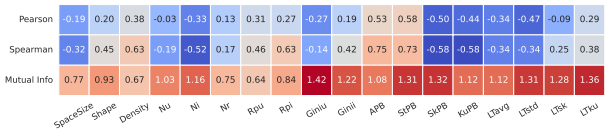


Figure 7: Pearson, Spearman correlations, and Mutual information between data characteristics and RecSys algorithms performance.

It appears that datasets exhibiting higher levels of popularity bias *APB* and *Density* tend to simplify the prediction tasks for recommender models. Conversely, datasets exhibit long-tailed item distributions, increased item diversity, and pronounced Popularity Bias, presenting a greater challenge for recommender models. Furthermore, the moderate mutual information values emphasize the practical impact of these characteristics on the performance of the models. To substantiate our findings, we have graphically depicted the *nDCG*@10 metrics across these features, illustrating how they vary alongside these characteristics in Figure 12.

4.4 Selection of benchmarking datasets

Using 30 public datasets in the benchmarking process, we need to justify that our datasets are quite informative to cover all recommendation datasets. Moreover, it would be right to give our benchmarking process an opportunity to select datasets which may belong to the same group. We use the KMeans approach to split datasets into multiple clusters. In this case, statistical characteristics [9] are selected as feature representations.

Principal datasets selection. To decrease time consumption and minimise the degradation of benchmarking, we can run it only for a limited number of datasets, carefully selecting them. We use several approaches for selection: Random, KMeans, A-optimality, and D-optimality approaches.

In Random, we uniformly at random select a subset of datasets from the set. The KMeans identifies core datasets as the closest ones to cluster centres for clusters being selected in the space of data characteristics [9]. Two additional baselines are A-optimality and D-optimality. They constitute two fundamental criteria focused on obtaining the lowest possible error of a model that predicts performance and the error for parameters estimation of a model [78]. Technical details are provided in Appendix C.

We collected 6 datasets and calculated Spearman's correlation between metrics for all datasets and 6 datasets. The correlations are an average of 500 runs. Results are presented in Table 4 for distinct evaluation metrics. KMeans approach outperforms any other

Method	nDCG10	HitRate10	Coverage
Random	0.834	0.855	0.939
D optimal	0.805	0.819	0.913
A optimal	0.669	0.687	0.887
KMeans	**0.845**	**0.900**	**0.982**

Table 4: Spearman correlation between metrics for six selected datasets and all 30 used datasets

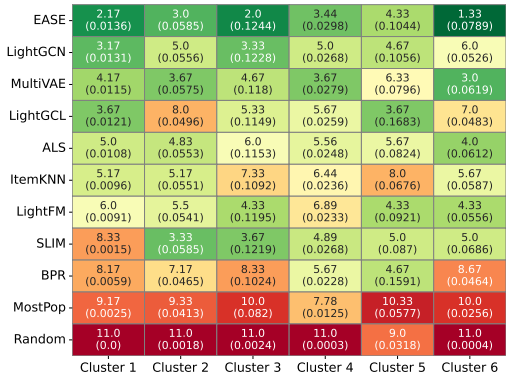


Figure 8: Ranks and geometric mean (in brackets) for nDCG10 on different clusters of datasets.

approach. It can be explained by Greedy Algorithms to optimize D optimal and A optimal approaches and limited performance for a regression model-based performance prediction and subsequent model mismatch.

Examination of clusters. With the clustering approach described above, we obtain separate ranking and metric values for different clusters. The top approach in almost all cases is EASE, losing only to LightGCL for Cluster 5. Other approaches are less stable. For example, LightGCL has an average rank of 8 for cluster 2, while SLIM has the second-best average rank for it. Part of this variety can be explained by the complexity of clusters: the geometric mean varies significantly from 0.0136 to 0.1244 for EASE. Another common thing is inferiority for MostPop and Random approaches, suggesting that for all considered datasets, learning makes sense.

5 CONCLUSIONS

Our paper introduces a novel benchmarking system for recommender systems. It integrates a clear pipeline that includes the usage of multiple datasets, tuning of hyperparameters and validation strategy, as well as an aggregation procedure for metrics across different datasets. Our approach is interpretable and robust, working for distinct metrics used for RecSys evaluation. Among the considered methods, we identify EASE as a clear winner with respect to all considered aggregation strategies. Other methods show inferior performance on average while being interesting for particular subdomains identified by our clustering scheme.

Further research provides deeper insight related to the stability and efficiency of ranking. Due to the usage of 30 datasets, two of which are open-sourced in this study, the results are robust in diverse considered scenarios. Via our clustering procedure, we obtain a collection of datasets of size 6 that also provides a consistent ranking, achieving efficiency and reliability simultaneously. Additional experiments confirm the stability of our benchmark with respect to reducing the number of considered datasets, methods, and adversarial manipulation of the list of methods. Overall, our research offers a streamlined guide and valuable datasets for advancing recommender system studies that can be used both by practitioners during the selection of a method and researchers during the evaluation of a novel idea.

REFERENCES

[1] Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. In *ACM SIGKDD*, pages 1748–1757, 2020.

[2] Aixin Sun. Take a fresh look at recommender systems from an evaluation standpoint. In *ACM SIGIR*, pages 2629–2638, 2023.

[3] Zhu Sun, 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 *ACM RecSys*, pages 23–32, 2020.

[4] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. A troubling analysis of reproducibility and progress in recommender systems research. *ACM TOIS*, 39(2):1–49, 2021.

[5] Balázs Hidasi and Ádám Tibor Czapp. The effect of third party implementations on reproducibility. In *ACM RecSys*, pages 272–282, 2023.

[6] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. Elliot: A comprehensive and rigorous framework for reproducible recommender systems evaluation. In *ACM SIGIR*, pages 2405–2414, 2021.

[7] Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, et al. Recbole 2.0: towards a more up-to-date recommendation library. In *ACM CIKM*, pages 4722–4726, 2022.

[8] Zhu Sun, Hui Fang, Jie Yang, Xinghua Qu, Hongyang Liu, Di Yu, Yew-Soon Ong, and Jie Zhang. DaisyRec 2.0: Benchmarking recommendation for rigorous evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[9] Jin Yao Chin, Yile Chen, and Gao Cong. The datasets dilemma: How much do we really know about recommendation datasets? In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 141–149, 2022.

[10] Alejandro Bellogín, Pablo Castells, and Iván Cantador. Statistical biases in information retrieval metrics for recommender systems. *Information Retrieval Journal*, 20:606–634, 2017.

[11] Mark Rofin, Vladislav Mikhailov, Mikhail Florinsky, Andrey Kravchenko, Tatiana Shavrina, Elena Tutubalina, Daniel Karabekyan, and Ekaterina Artemova. Vote'n'rank: Revision of benchmarking with social choice theory. In *EACL*, pages 670–686, 2023.

[12] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *ACM conference on CSCW*, pages 175–186, 1994.

[13] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Conference on WWW*, pages 285–295, 2001.

[14] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[15] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM CSUR*, 52(1):1–38, 2019.

[16] Wen Zhang, Yuhang Du, Taketoshi Yoshida, and Ye Yang. DeepRec: A deep neural network approach to recommendation with item embedding and weighted loss function. *Information Sciences*, 470:121–140, 2019.

[17] Implicit: Fast python collaborative filtering for implicit datasets. https://github.com/benfred/implicit. Accessed: 2016.

[18] Maciej Kula. Metadata embeddings for user and item cold-start recommendations. In Toine Bogers and Marijn Koolen, editors, *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015)*, volume 1448 of *CEUR Workshop Proceedings*, pages 14–21. CEUR-WS.org, 2015.

[19] Bin Wu, Zhongchuan Sun, He Xiangnan, Xiang Wang, and Jonathan Staniforth. NeuRec: An open source neural recommender library. https://github.com/wubinzzu/NeuRec. Accessed: 2020.

[20] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *ACM CIKM*, pages 4653–4664, 2021.

[21] Lien Michiels, Robin Verachtert, and Bart Goethals. Recpack: An(other) experimentation toolkit for top-n recommendation using implicit feedback data. In *ACM RecSys*, RecSys '22, page 648–651, New York, NY, USA, 2022. Association for Computing Machinery.

[22] Alexey Vasilev. Replay: A library for building recommender system models using pyspark. https://github.com/sberbank-ai-lab/RePlay. Accessed: 2020.

[23] Wayne Xin Zhao, Zihan Lin, Zhichao Feng, Pengfei Wang, and Ji-Rong Wen. A revisiting study of appropriate offline evaluation for top-n recommendation algorithms. *ACM Transactions on Information Systems*, 41(2):1–41, 2022.

[24] Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, and Felice Antonio Merra. How dataset characteristics affect the robustness of collaborative recommendation models. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 951–960, 2020.

[25] Yashar Deldjoo, Alejandro Bellogin, and Tommaso Di Noia. Explaining recommender systems fairness and accuracy through the lens of data characteristics. *Information Processing & Management*, 58(5):102662, 2021.

[26] Pierre Jean A Colombo, Chloé Clavel, and Pablo Piantanida. Infolm: A new metric to evaluate summarization & data2text generation. In *AAAI Conference on Artificial Intelligence*, volume 36, pages 10554–10562, 2022.

[27] Christina Nießl, Moritz Herrmann, Chiara Wiedemann, Giuseppe Casalicchio, and Anne-Laure Boulesteix. Over-optimism in benchmark studies and the multiplicity of design and analysis options when interpreting their results. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 12(2):e1441, 2022.

[28] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91:201–213, 2002.

[29] Mikhail Belyaev, Evgeny Burnaev, Ermek Kapushev, Maxim Panov, Pavel Prikhodko, Dmitry Vetrov, and Dmitry Yarotsky. GTApprox: Surrogate modeling for industrial design. *Advances in Engineering Software*, 102:29–39, 2016.

[30] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006.

[31] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. Hive-cote 2.0: a new meta ensemble for time series classification. *Machine Learning*, 110(11-12):3211–3243, 2021.

[32] Alessio Benavoli, Giorgio Corani, and Francesca Mangili. Should we really use post-hoc tests based on mean-ranks? *JMLR*, 17(1):152–161, 2016.

[33] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in Statistics: Methodology and Distribution*, pages 196–202. Springer, 1992.

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[35] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi, editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

[36] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *NeurIPS*, 32, 2019.

[37] Pieter Gijsbers, Marcos LP Bueno, Stefan Coors, Erin LeDell, Sébastien Poirier, Janek Thomas, Bernd Bischl, and Joaquin Vanschoren. AMLB: an AutoML benchmark. *arXiv:2207.12560*, 2022.

[38] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, Dietmar Jannach, and Claudio Pomo. Top-N recommendation algorithms: A quest for the state-of-the-art. In *ACM UMAP*, pages 121–131, 2022.

[39] Jieming Zhu, Quanyu Dai, Liangcai Su, Rong Ma, Jinyang Liu, Guohao Cai, Xi Xiao, and Rui Zhang. BARS: Towards open benchmarking for recommender systems. In *ACM SIGIR*, pages 2912–2923, 2022.

[40] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM TIIS*, 5(4), dec 2015.

[41] James Bennett, Stan Lanning, et al. The Netflix prize. In *KDD cup and workshop*, volume 2007, page 35. New York, 2007.

[42] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. Recommender systems with social regularization. In *ACM WSDM*, WSDM '11, pages 287–296, Hong Kong, China, 2011.

[43] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP-IJCNLP*, pages 188–197, 2019.

[44] Chongming Gao, Shijun Li, Wenqiang Lei, Jiawei Chen, Biao Li, Peng Jiang, Xiangnan He, Jiaxin Mao, and Tat-Seng Chua. Kuairec: A fully-observed dataset and insights for evaluating recommender systems. In *ACM CIKM*, CIKM '22, page 540–550, 2022.

[45] Rekko Dataset competition. https://www.kaggle.com/datasets/g0ohard/rekko-challenge. Accessed: 2023-10-9.

[46] Julian McAuley, Jure Leskovec, and Dan Jurafsky. Learning attitudes and attributes from multi-aspect reviews. In *IEEE ICDM*, pages 1020–1025. IEEE, 2012.

[47] RateBeer Dataset competition. https://www.kaggle.com/datasets/ankurnapa/rate-beer-data. Accessed: 2023-10-9.

[48] Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. Generating Personalized Recipes from Historical User Preferences. In *EMNLP-IJCNLP*, pages 5976–5982, Hong Kong, China, November 2019. Association for Computational Linguistics.

[49] Ta Feng Grocery Dataset competition. https://www.kaggle.com/datasets/chiranjivdas09/ta-feng-grocery-dataset. Accessed: 2023-10-9.

[50] Yelp Dataset competition. https://www.yelp.com/dataset. Accessed: 2023-10-9.

[51] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. Trust management for the semantic web. In *ISWC*, pages 351–368. Springer, 2003.

[52] Srijan Kumar, William L Hamilton, Jure Leskovec, and Dan Jurafsky. Community interaction and conflict on the web. In *Conference on WWW*, pages 933–943. International World Wide Web Conferences Steering Committee, 2018.

[53] Hui Li, Dingming Wu, Wenbin Tang, and Nikos Mamoulis. Overlapping community regularization for rating prediction in social recommender systems. In *RecSys*, pages 27–34, 2015.

[54] MTS Library Dataset competition. https://www.kaggle.com/datasets/sharthz23/mts-library. Accessed: 2023-10-9.

[55] Mengting Wan and Julian J. McAuley. Item recommendation on monotonic behavior chains. In Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O'Donovan, editors, *ACM RecSys*, pages 86–94. ACM, 2018.

[56] Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian J. McAuley. Fine-grained spoiler detection from large-scale review corpora. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *ACL*, pages 2605–2610. Association for Computational Linguistics, 2019.

[57] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *ACM SIGKDD*, pages 1082–1090, 2011.

[58] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhiwen Yu. Fine-grained preference-aware location search leveraging crowdsourced digital footprints from lbsns. In *ACM UBICOMP*, pages 479–488, 2013.

[59] Jacek Dąbrowski, Barbara Rychalska, Michał Daniluk, Dominika Basaj, Konrad Gołuchowski, Piotr Bąbel, Andrzej Michałowski, and Adam Jakubowski. An efficient manifold density estimator for all recommendation systems. In *ICONIP*, pages 323–337. Springer, 2021.

[60] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *IEEE ICDM*, pages 263–272. Ieee, 2008.

[61] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[62] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *IEEE ICDM*, pages 497–506. IEEE, 2011.

[63] Harald Steck. Embarrassingly shallow autoencoders for sparse data. In *Conference on WWW*, pages 3251–3257, 2019.

[64] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Conference on WWW*, pages 689–698, 2018.

[65] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 639–648, New York, NY, USA, 2020. Association for Computing Machinery.

[66] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. Lightgcl: Simple yet effective graph contrastive learning for recommendation. 2023.

[67] Pablo Castells and Alistair Moffat. Offline recommender system evaluation: Challenges and new directions. *AI Magazine*, 43(2):225–238, 2022.

[68] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. A critical study on data leakage in recommender system offline evaluation. *ACM Transactions on Information Systems*, 41(3):1–27, 2023.

[69] Pedro G Campos, Fernando Díez, and Iván Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24:67–119, 2014.

[70] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. Exploring data splitting strategies for the evaluation of recommendation models. In *ACM RecSys*, pages 681–686, 2020.

[71] Longqi Yang, Yin Cui, Yuan Xuan, Chenyang Wang, Serge Belongie, and Deborah Estrin. Unbiased offline recommender evaluation for missing-not-at-random implicit feedback. In *ACM RecSys*, pages 279–287, 2018.

[72] Yan-Martin Tamm, Rinchin Damdinov, and Alexey Vasilev. Quality metrics in recommender systems: Do we calculate metrics consistently? In *ACM RecSys*, pages 708–713, 2021.

[73] Marius Kaminskas and Derek Bridge. Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM TiiS*, 7(1):1–42, 2016.

[74] Tobias Schnabel. Where do we go from here? guidelines for offline recommender evaluation. *arXiv preprint arXiv:2211.01261*, 2022.

[75] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *ACM SIGKDD*, pages 2623–2631, 2019.

[76] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Claudio Pomo, and Azzurra Ragone. On the discriminative power of hyper-parameters in cross-validation and how to choose them. In *ACM RecSys*, pages 447–451, 2019.

[77] Alan Said and Alejandro Bellogín. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *ACM RecSys*, pages 129–136, 2014.

[78] Friedrich Pukelsheim. *Optimal design of experiments*. SIAM, 2006.

[79] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. 2011.

[80] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[81] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

A DATASETS

A.1 Datasets

One significant drawback of conventional benchmarking methods is the limited availability of datasets for thorough evaluation. To ensure robust and reliable results with a predefined level of significance, having a larger pool of datasets is advantageous. In light of this, our model evaluation incorporate 30 diverse datasets spanning various domains and structures, as detailed in Table 1. These datasets can be broadly categorized into the following structures:

- *Classic interactions* category encompasses conventional user-item interactions, whether they are implicit or explicit. In cases of explicit interactions, we consider the rating as a measure of interaction intensity.
- *Session-based interactions* relates to user group activities occurring within a single session. What sets this category apart is the categorization of events and the potential for event repetition, allowing for recurring user-item pairings. In our analysis, we aggregate recurring events with appropriate measures of intensity.
- *Network-based* category primarily emerges within social networks, where we interpret connections as interactions.

Expanding the scope of our research, we introduce two new datasets, each sourced from distinct sectors necessitating recommendation systems (music and e-commerce). Cumulatively, our analysis covers a set of 30 datasets.

A.2 New datasets

Additionally, we present two new datasets, that we call Dataset 1 and Dataset 2.

Dataset 1 chronicles user behavior through events such as views, favorites, additions to the cart, and purchases captured over the span of five months from January 15 to May 15, 2023. It encompasses a total of 196,644,020 events, spanning 3,562,321 items across 10,001 distinct categories, contributed by 2,730,776 unique users.

Content:

- **User IDs**: Distinct identifiers for users, totaling 2,730,776.
- **Datetimes**: Timestamp range from 01.15.2023 00:00:00.708 to 14.05.2023 20:59:59.000.
- **Events**: Categorized by unique codes, comprised of:
  - 0: View
  - 1: Favorites
  - 2: Add to cart
  - 3: Purchase
- **Item IDs**: Specific identifiers for items, amounting to 3,562,321.
- **Category IDs**: Denoting which of the 10,001 unique categories an item belongs to.
- **Prices**: Prices of items, normalized to follow a $\mathcal{N}(0, 1)$ distribution.

Dataset 2 captures users' song-listening experiences over the same five-month period. It records 244,673,551 events, taking place over 12,598,314 listening sessions. These sessions were initiated by 382,790 unique users and spanned 1,506,950 individual tracks. Notably, this dataset is exclusive to music and does not include other auditory content like podcasts or audiobooks.

Content:

- **User IDs**: Individual identifiers, with a count of 382,790 users.
- **Session IDs**: IDs for users' listening sessions, totaling 12,598,314.
- **Datetimes**: Timestamps spanning from 01.15.2023 to 14.05.2023.
- **Track IDs**: Identifiers for the music tracks, encompassing 1,506,950 unique tracks.
- **Play Durations**: Scaled durations of tracks played, considering tracks where at least 30% of the song's duration was completed.

A.3 Preprocessing

In our research, we concentrate solely on collaborative filtering, which requires the transformation of datasets into implicit feedback. To achieve this, we utilize threshold binarization. For the majority of datasets, where explicit rating values range from 0 to 5, we set a positive threshold at 3.5. Moreover, certain datasets involve weights within the range of 0 to 1, such as datasets with percentage-type data (e.g., book reading progress). For these datasets, we establish a positive threshold of 0.3. For the remaining datasets, we determine thresholds based on the drop ratio.

In addition to rating data, certain datasets, such as Dataset 1, reflect user behavior through specific events. These present challenges, primarily due to the presence of repeated user-item rating pairs, which are incompatible with collaborative filtering. To address this, we preprocess the data in the following manner:

(1) Assign weights using the formula:

$$\frac{\sum \text{all\_interactions}}{\sum \text{type\_interactions}} \tag{1}$$

where *type_interactions* denotes the count of events of a specific type. Thus, rarer events receive higher weights.

(2) Aggregate events by pairing users and items, retaining only the most frequent event type. We establish weight boundaries to ensure that events of a lesser significance never outnumber those of greater importance.

(3) Treat the newly assigned weights as ratings and apply threshold binarization accordingly.

This preprocessing approach ensures consistent dataset handling in line with the principles of collaborative filtering.

B RECOMMENDATION ALGORITHMS

In this study, we consider eleven baseline methods for our analysis. **Random** and **MostPop** are non-personalized methods that recommend random and most popular items to all users, respectively. **ItemKNN** are K-nearest neighborhood-based method that provide recommendations based on item similarity. Alternating Least Squares (**ALS**) performs a weighted matrix factorization, while Bayesian Personalized Ranking (**BPR**) learns pairwise personalized rankings from users' implicit feedback data. **LightFM** with Weighted Approximate-Rank Pairwise (WARP) [79] loss samples positive and negative items and running pairwise comparisons. **SLIM** and **EASE** learn linear functions to capture item-based collaborative filtering similarity, with SLIM aiming to learn a sparse linear

	Parameter	Searching Space	Description
ItemKNN	k	[10; 2500]	the neighborhood size
	shrink	[0.0; 1.0]	a normalization parameter
ALS	factors	[50; 200]	the number of latent factors
	regularization	[1e-4; 10]	the regularization factor
	iterations	[1; 100]	the number of ALS iterations
	alpha	[1; 50]	the weight to give to positive examples
BPR	factors	[50; 200]	the number of latent factors
	regularization	[1e-4; 10]	the regularization factor
	iterations	[10; 100]	the number of training epochs
	learning rate	[1e-3; 1e-1]	
LightFM	loss	warp	the loss function
	no. components	[10; 100]	the dimensionality of latent embeddings
	max sampled	[10; 50]	max negative samples used during WARP fitting
	learning rate	[1e-5; 1e-2]	
SLIM	alpha	[0.0; 1.0]	control the weights of L1 and L2 norms
	L1 ratio	[0.01; 0.5]	control the weights of L1 and L2 norms
EASE	reg. weight	[1.0; 1000.0]	the L2 regularization weight
MultiVAE	latent dimension	[32; 256]	the latent dimension of auto-encoder
	anneal cap	[0.0; 1.0]	weight of KL loss
	dropout	[0.0; 0.7]	the dropout probability
	learning rate	[1e-5; 1e-2]	
LightGCL	embedding size	[32; 256]	dimension of node embeddings
	q	[3; 10]	control parameter for negative samples
	n_layers	[1; 4]	number of convolutional layers
	learning rate	[1e-5; 1e-2]	learning rate for optimization
	lambda1	[1e-7; 0.3]	weight for L1 regularization
	lambda2	[1e-8; 1e-4]	weight for L2 regularization
	temp	[0.1; 5]	temperature parameter
	dropout	{0.0, 0.1, 0.25}	dropout rate for model
LightGCN	embedding size	[32; 256]	dimension of user and item embeddings
	n_layers	[1; 4]	number of graph convolutional layers
	reg weight	[1e-5; 1e-3]	regularization weight
	learning rate	[1e-4; 1e-2]	learning rate for optimization

Table 5: The hyperparameter search space.

function, while EASE does not have the sparsity constraint. **MultiVAE** extends variational and denoising autoencoders to collaborative filtering using a multinomial likelihood. **LightGCN** adapts simplified version of GCN [80, 81] to recommendation by using only neighborhood aggregation — for collaborative filtering. **LightGCL** improves GNN-based methods by implementing a contrastive learning framework, which causes robustness against data sparsity and popularity bias. The search space for the hyperparameters associated with these baselines is provided in Table 5.

C SELECTION OF THE OPTIMAL SUBSET OF DATASETS

In the KMeans clustering process, we first standardize the data to ensure uniformity. Next, we apply Principal Component Analysis (PCA) to reduce the number of dimensions while preserving as much variance as possible, addressing the problem of correlated features. We then use the Isolation Forest method to detect and remove outliers, decreasing the dataset size from 30 to 25 observations. With the data now prepared, we move on to the clustering stage, employing the K-means algorithm. During clustering, we evaluate the Silhouette Coefficient and Davies-Bouldin Scores to determine the optimal number of clusters, which we found to be 6. Finally, we select datasets that are closest to the cluster centres for further analysis.

The next two approaches assume that resulting metrics can be predicted with the linear regression method:

$$\mathbf{y} = \mathbf{x}\Theta + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2),$$

where $\mathbf{y}$ is metric on datasets, Θ means model parameters, $\mathbf{x}$ means datasets feature and ε is a noise value.

The Maximum Likelihood Estimation weights can be shown as

$$\Theta \sim \mathcal{N}\left((\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^{\mathbf{x}}\Theta, (\frac{1}{\sigma^2}\mathbf{x}^T\mathbf{x})^{-1}\right)$$

The D-optimality approach we formulate as maximization of $\mathbf{x}^T\mathbf{x}$ determinant. As a result, we minimize the variance of the parameter.

We formulate the A-optimality as a result of the minimization of mean model loss. In this case, we assume that the prior distribution is standard normal distribution $p(x) = \mathcal{N}(0, I)$

$$Q(\mathcal{D}) = \int (y(x) - \hat{y}(x))^2 p(x)dx = \frac{1}{3}\text{tr}((\mathbf{x}^T\mathbf{x})^{-1})$$

These problems are discrete optimization, and the naive solution is a Greedy algorithm. After selecting the starting datasets' subset, we perform a complete search across each dataset, choosing datasets with high values of the target function.

D EVALUATION METRICS

The details of each metric are presented in this section using the following notations:

- u is a user identification
- i is an item identification
- N_k^{rec} is a top-k set of items in the recommendations. $N_k^{rec}(u)$ denotes the set of recommendations for the user u.
- N^{rel} is a set of the relevant items. $N^{rel}(u)$ correspond to the user u.
- N^t is a set of the history items.
- M is a set of users.
- *rank(u,i)* is a position of item i in the recommendation list for the user u.
- $\mathbb{1}[\cdot]$ is an indicator function.

The rest of the designations are designated further as they appear.

Precision and Recall. The most common definitions of these metrics are the following:

$$Precision@k = \frac{1}{|M|}\sum_{u \in M}\frac{|N^{rel}(u) \cap N_k^{rec}(u)|}{k} \tag{2}$$

$$Recall@k = \frac{1}{|M|}\sum_{u \in M}\frac{|N^{rel}(u) \cap N_k^{rec}(u)|}{|N^{rel}(u)|} \tag{3}$$

A potential issue arises when a user has fewer relevant items in the test data than the specified value of k. In such cases, the metric's normalization is compromised because the optimal outcome is not fixed at 1; it can vary. To address this, we utilize the concept of **modified k**.

$$k_m(u) = \min(k, |N^{rel}(u)|) \tag{4}$$

Therefore, in our case *Precision@k* metric will be the following:

$$Precision@k = \frac{1}{|M|}\sum_{u \in M}\frac{|N^{rel}(u) \cap N_k^{rec}(u)|}{k_m(u)} \tag{5}$$

Mean Average Precision and normalized Discounted Cumulative Gain. While the aforementioned metrics focus on accuracy, they overlook the order of predictions. Even when the value of k is low, it's important to differentiate between models with better ranking capabilities.

Average precision takes into account the order of predicted items:

$$AP@k(u) = \frac{1}{k_m(u)} \sum_{i \in N_k^{rec}(u)} \mathbb{1}[i \in N^{rel}(u)] Precision@rank(u,i)(u) \tag{6}$$

Mean average precision simply averages this equation across the users:

$$MAP@k = \frac{1}{|M|} \sum_{u \in M} AP@k(u) \tag{7}$$

Another metric is *nDCG@k*:

$$nDCG@k(u) = \frac{DCG@k(u)}{iDCG@k(u)}, \tag{8}$$

where

$$DCG@k(u) = \sum_{i \in N_k^{rec}} \frac{\mathbb{1}[i \in N^{rel}(u)]}{\log_2(rank(u,i)+1)} \tag{9}$$

iDCG@k represents the biggest possible value of *DCG@k*:

$$iDCG@k(u) = \sum_{k=1}^{k_m(u)} \frac{1}{\log_2(k+1)} \tag{10}$$

Mean Reciprocal Rank. *MRR@k* is the mean of the inverse position of the first relevant item.

$$MRR@k(u) = \frac{1}{\min\limits_{i \in N^{rel}(u) \cap N_k^{rec}(u)} rank(u,i)} \tag{11}$$

If no items were predicted correctly, it is defined as 0.

HitRate. *HitRate@k* is the softest metric. it is defined as 1 if at least one item in the recommendation list was relevant and 0 otherwise.

$$HitRate@k(u) = \mathbb{1}[|N^{rel}(u) \cap N_k^{rec}(u)| > 0] \tag{12}$$

Coverage. The previous metrics assessed quality for individual users. These metrics evaluate model diversity and novelty. *Coverage@k* is the proportion of recommended items that appear in all users' recommendations.

$$Coverage@k = \frac{|N^{rec}|}{|N^t|}. \tag{13}$$

Diversity. The presence of different types within a set of recommendations is referred to as *Diversity@k*. *Intra-List Similarity* (*IL@k*) calculates the similarity between all items in a group. If a recommendation list has many similar items, the *IL@k* score will be high. Conversely, a lower score indicates greater diversity. *Cosine similarity* is utilized to measure the similarity between two items.

$$\text{Cosine Similarity(i, j)} = \frac{|M_t^{l(i,j)}|}{\sqrt{|M_t^{l(i)}|}\sqrt{|M_t^{l(j)}|}}, \tag{14}$$

where $M_t^{l(i)}$ denotes the set of users who liked item i and $M_t^{l(i,j)}$ the users who liked both i and j. If user-item interactions can be represented as a binary sparse matrix, with the users on the rows and items on the columns, the upper equation transforms into this:

$$\text{Cosine Similarity(i, j)} = \frac{c_i c_j}{\|c_i\|\|c_j\|}, \tag{15}$$

where c_i represents the i-th column of the binary interaction matrix. The *IL@k* is then defined as

$$IL@k = \frac{1}{|M|} \sum_{u \in M} \frac{1}{\binom{N_k^{rec}(u)}{2}} \sum_{i,j \in N_k^{rec}(u), i<j} \text{Cosine Similarity(i, j)}. \tag{16}$$

Finally, diversity is defined as

$$Diversity@k = 1 - IL@k. \tag{17}$$

Novelty. The more popular an item is, the less novel it is considered to be. This relationship can be expressed through the probability of an item i being observed or interacted with by users, denoted as *p(i)*.

$$p(i) = \frac{|M_t^{l(i)}|}{|N^t|} \tag{18}$$

The novelty of an item is then defined as

$$novelty(i) = -\log_2 p(i) \tag{19}$$

And the *Novelty@k* of the recommendations across all users is defined as

$$Novelty@k = \sum_{i \in N_k^{rec}} \frac{|M_k^r(i)|}{|N^{rec}|} novelty(i), \tag{20}$$

where M_k^r denotes the users who were recommended item i in the top-k predictions.

E EVALUATION RESULTS

Results of the evaluation of the models are demonstrated within one of the most popular RecSys datasets – Movielens_1m. Metrics can be viewed in the Table 6.

F ADDITIONAL EXPERIMENTS

F.1 Critical difference diagram for small number of datasets

We compare the CD diagram for 30 datasets with the CD diagram for 5 uniformly selected datasets. The latter, as shown in Figure 9, indicates that while mean ranks, especially among the top approaches, are similar, the differences are statistically significant. This suggests the necessity of more datasets to draw confident conclusions.

Method	k	Precision@k	Recall@K	MAP@K	nDCG@k	MRR@k	HitRate@k	Coverage@k	Diversity@k	Novelty@k
EASE	5	0.182538	0.033129	0.124235	0.187298	0.308818	0.484822	0.126720	0.623048	4.190139
	10	0.182852	0.060302	0.100707	0.183565	0.327504	0.622723	0.169920	0.651442	4.270349
	20	0.188212	0.102458	0.082286	0.179556	0.334797	0.726800	0.233600	0.685269	4.382420
	100	0.320327	0.304552	0.073958	0.224839	0.339885	0.918474	0.456640	0.769821	4.723435
MultiVAE	5	0.193380	0.034838	0.135458	0.198792	0.318835	0.491761	0.180160	0.605884	4.190265
	10	0.190887	0.059817	0.109959	0.193018	0.336969	0.619748	0.240640	0.633229	4.279846
	20	0.187216	0.101157	0.087728	0.185392	0.344192	0.728517	0.321280	0.662076	4.388697
	100	0.316456	0.300759	0.071485	0.221170	0.348395	0.914063	0.624320	0.748402	4.731920
ItemKNN	5	0.199375	0.035284	0.144187	0.205892	0.329150	0.492084	0.192000	0.618495	4.191397
	10	0.186842	0.061476	0.113796	0.190234	0.346383	0.624311	0.256000	0.645890	4.280308
	20	0.182639	0.103982	0.089335	0.181671	0.353607	0.734796	0.339200	0.674737	4.389258
	100	0.311289	0.295103	0.072997	0.217053	0.357810	0.910342	0.672000	0.761063	4.732481
SLIMElastic	5	0.190675	0.033752	0.130239	0.196364	0.317019	0.485945	0.203520	0.630046	4.192529
	10	0.188183	0.059431	0.104740	0.190590	0.335153	0.623905	0.272640	0.657500	4.281440
	20	0.185472	0.100087	0.085521	0.182027	0.342377	0.731390	0.361760	0.686347	4.390391
	100	0.314122	0.292431	0.069678	0.217409	0.346580	0.907937	0.706880	0.772712	4.733614
LightGCN	5	0.205678	0.036829	0.148267	0.213780	0.339361	0.503307	0.214880	0.637597	4.193661
	10	0.192245	0.062489	0.117835	0.198102	0.356595	0.631452	0.286720	0.665051	4.282572
	20	0.188043	0.104692	0.091422	0.189539	0.363819	0.738937	0.379840	0.693898	4.391523
	100	0.316873	0.289775	0.075579	0.224921	0.368022	0.915484	0.731360	0.780263	4.734746
LightGCL	5	0.207989	0.037341	0.150268	0.215780	0.341562	0.505500	0.225280	0.640098	4.194793
	10	0.194556	0.063510	0.119836	0.199402	0.358796	0.633645	0.300640	0.667552	4.283704
	20	0.190354	0.105713	0.093423	0.190839	0.366020	0.741130	0.395760	0.696399	4.392655
	100	0.319184	0.291068	0.077580	0.226221	0.370223	0.917677	0.750880	0.782764	4.735878
ALS	5	0.183283	0.030528	0.126192	0.186707	0.299763	0.476673	0.716800	0.567343	100.872475
	10	0.180620	0.056412	0.101041	0.181550	0.317732	0.607505	0.824320	0.599500	168.448549
	20	0.187109	0.097356	0.084409	0.179479	0.324959	0.710953	0.858240	0.634799	289.977511
	100	0.320542	0.304543	0.074687	0.225015	0.330163	0.903651	0.877440	0.733591	1034.915677
BPR	5	0.184009	0.027342	0.133611	0.192481	0.314300	0.466531	0.696640	0.482689	245.131216
	10	0.169465	0.045537	0.103246	0.178866	0.329334	0.577079	0.882240	0.530301	404.959665
	20	0.169042	0.081060	0.081759	0.171175	0.337496	0.697769	0.969280	0.570537	643.367459
	100	0.267711	0.252355	0.063515	0.196756	0.342227	0.880325	0.987520	0.714399	2191.723079
LightFM	5	0.192241	0.032208	0.133199	0.195460	0.311072	0.481744	0.280000	0.516479	120.068541
	10	0.190753	0.055784	0.110381	0.191948	0.329554	0.618661	0.382400	0.545503	190.527897
	20	0.188639	0.093892	0.090419	0.184581	0.336217	0.711968	0.501120	0.585300	316.817541
	100	0.298608	0.281632	0.075019	0.218924	0.340421	0.882353	0.757440	0.701994	1106.280685
MostPop	5	0.181356	0.026874	0.128595	0.185752	0.292022	0.448276	0.301760	0.468761	303.670962
	10	0.165174	0.044425	0.099545	0.172147	0.307775	0.562880	0.414080	0.504850	477.119862
	20	0.167009	0.078456	0.080757	0.166928	0.316198	0.682556	0.544320	0.549585	735.897483
	100	0.272350	0.256049	0.064653	0.197877	0.321170	0.873225	0.804160	0.679447	2228.499972
Random	5	0.012576	0.001737	0.005615	0.012192	0.026386	0.060852	0.761920	0.953472	56.162820
	10	0.015890	0.003644	0.004350	0.014613	0.036291	0.135903	0.927360	0.952976	112.209682
	20	0.017111	0.007026	0.003164	0.015227	0.041819	0.218053	0.990400	0.952594	224.647352
	100	0.035218	0.032637	0.002283	0.022971	0.049214	0.538540	1.000000	0.952413	1124.003685

Table 6: The results for Movielens 1m dataset.

F.2 Reliability tests: the incorporation or exclusion of slightly superior/inferior method

In this section, we investigate the effects of introducing a new model that closely resembles chosen models. We select a specific model and a parameter, denoted as α. The metrics for the new model are derived by multiplying the metrics of the chosen model by α. We have limited the parameter as follows: $|1 - \alpha| \leq 0.15$. For example, the case when $\alpha = 1.$ means adding the new model equal to the selected one.

By adjusting the parameter α, we manipulate the new model to perform either better or worse than the selected model. The outcomes of this investigation are illustrated in Figure 10a. We have omitted the Minimax aggregation method due to significantly poorer Spearman correlation results compared to other methods. It

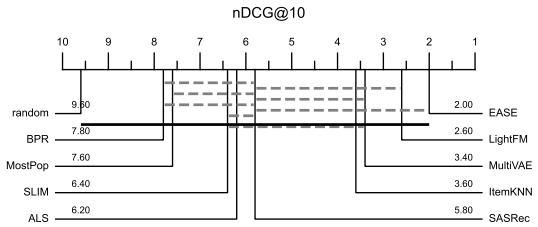


Figure 9: Critical Difference diagram constructed using five datasets.

is apparent that the majority of aggregation methods remain relatively stable, with the exception of the **Mean ranks** and **Copeland** methods.

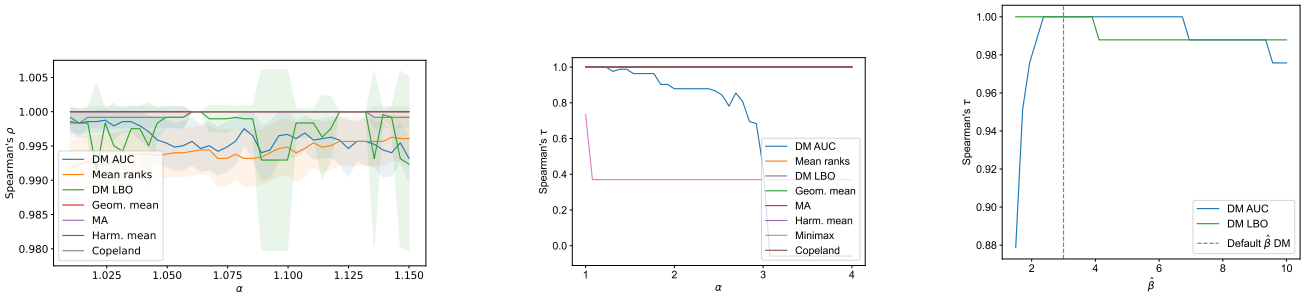
F.3 Reliability tests: add a new best method

In this section, we analyze the impact of introducing a new best model. We identify the best metric values across all datasets and utilize an arbitrary value α (following similar conditions as in Section F.2). We restrict the parameter as follows: $\alpha \in [1, 4]$. For instance, when $\alpha = 1$, it signifies the addition of a new model with metrics equal to the current best metrics.

The results of this investigation are depicted in Figure 10b. Notably, the **Minimax** aggregation method displays instability for any values of α. Additionally, the **DM AUC** method demonstrates significant instability as $\alpha \longrightarrow 4 - 0$. This behavior can be attributed to the direct dependence of the best metric values on the parameter α. Conversely, all other aggregation methods remain entirely stable in this scenario.

F.4 Sensitivity to hyperparameters of aggregations

The results for varying the hyperparameters case are presented in Figure 10c. The vertical dotted line mean the case when $\hat{\beta} = 3$. We see that the *DM AUC* aggregating method is more stable than the *DM LBO* method. Instability of the *DM AUC* can be interpreted by the proximity of the curves of the methods (in Figure 3, for example, *LightFM AUC* and *MultiVAE AUC* curves look visually similar to each other). If the more methods have similar perfomance curves, then the more the Spearman correlation value decreases (in Figure 3, for example, *LightFM AUC*, *MultiVAE AUC* and **ItemKNN AUC** curves for $\hat{\beta} \longrightarrow 1. + 0$).



(a) Comparison of stability of aggregations with respect to a new similar method

(b) Comparison of stability of aggregations with respect to the new best method

(c) Comparison of DM aggregation approaches via Spearman correlation across different hyperparameters

Figure 10: Reliability evaluation for various aggregation methods

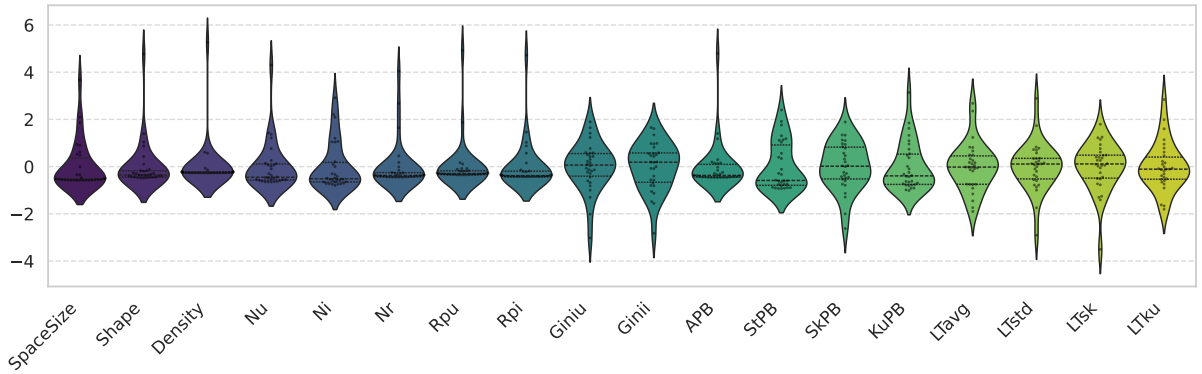


Figure 11: Variations in characteristics of datasets.

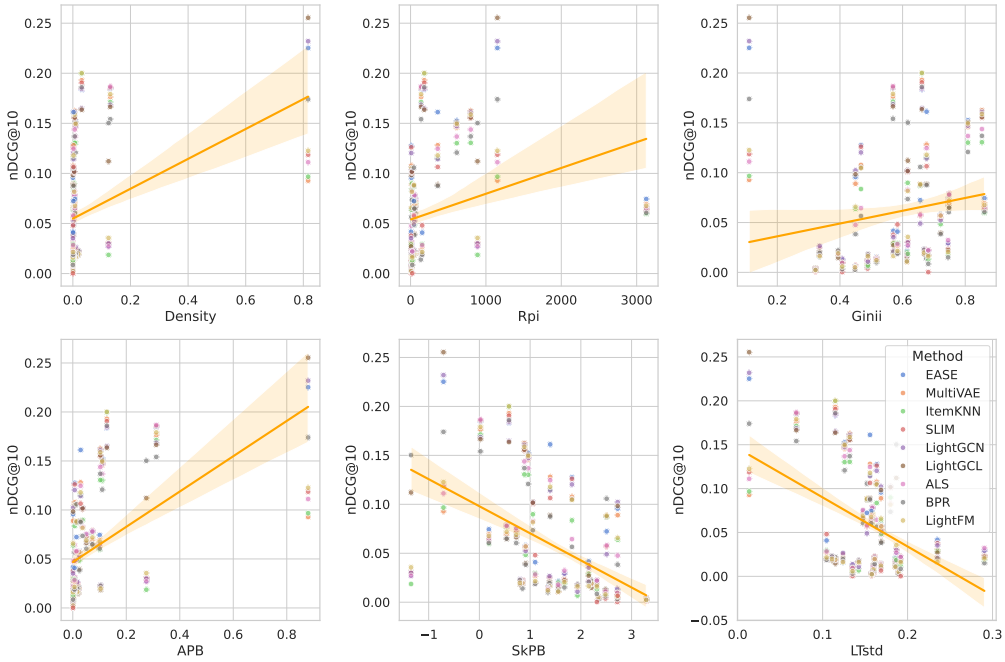


Figure 12: Variations in characteristics of datasets and corresponding values of nDCG@10 for RecSys algorithms.