The Effect of Random Seeds for Data Splitting on Recommendation Accuracy

Lukas Wegmeth[1], Tobias Vente[1], Lennart Purucker[1] and Joeran Beel[1]

[1]*Intelligent Systems Group, University of Siegen, Germany*

Abstract

The evaluation of recommender system algorithms depends on randomness, e.g., during randomly splitting data into training and testing data. We suspect that failing to account for randomness in this scenario may lead to misrepresenting the predictive accuracy of recommendation algorithms. To understand the community's view of the importance of randomness, we conducted a paper study on 39 full papers published at the ACM RecSys 2022 conference. We found that the authors of 26 papers used some variation of a holdout split that requires a random seed. However, only five papers explicitly repeated experiments and averaged their results over different random seeds. This potentially problematic research practice motivated us to analyze the effect of data split random seeds on recommendation accuracy. Therefore, we train three common algorithms on nine public data sets with 20 data split random seeds, evaluate them on two ranking metrics with three different ranking cutoff values k, and compare the results. In the extreme case with $k = 1$, we show that depending on the data split random seed, the accuracy with traditional recommendation algorithms deviates by up to ~6.3% from the mean accuracy achieved on the data set. Hence, we show that an algorithm may significantly over- or under-perform when maliciously or negligently selecting a random seed for splitting the data. To showcase a mitigation strategy and better research practice, we compare holdout to cross-validation and show that, again, for $k = 1$, the accuracy of algorithms evaluated with cross-validation deviates only up to ~2.3% from the mean accuracy achieved on the data set. Furthermore, we found that the deviation becomes smaller the higher the value of k for both holdout and cross-validation.

Keywords

recommender systems, random seed, holdout, validation, cross-validation, ranking, reproducibility

1. Introduction

Finding the best algorithm for a recommendation task is challenging, as the experimental setup requires many difficult design choices [1]. On a high level, these design choices include or are affected by: the preprocessing of input data, the set of available algorithms and hyperparameters, the evaluation metrics, the testing environment, and constraints on the experiment. Understanding the effects of every choice in each step of the experimental pipeline is crucial in obtaining a result that accurately represents the pipeline's capabilities. Especially in state-of-the-art papers, the evaluation procedure is often complex and lengthy, and sometimes the

Perspectives on the Evaluation of Recommender Systems Workshop (PERSPECTIVES 2023), September 19th, 2023, co-located with the 17th ACM Conference on Recommender Systems, Singapore, Singapore.

lukas.wegmeth@uni-siegen.de (L. Wegmeth); tobias.vente@uni-siegen.de (T. Vente); contact@lennart-purucker.com (L. Purucker); joeran.beel@uni-siegen.de (J. Beel)
0000-0001-8848-9434 (L. Wegmeth); 0009-0003-8881-2379 (T. Vente); 0009-0001-1181-0549 (L. Purucker); 0000-0002-4537-5573 (J. Beel)

performance increase over competing solutions is small [2, 3, 4]. Notably, many components in a recommender systems evaluation pipeline require randomness, e.g., to split the data or initialize random parameters for an algorithm.

Any function that generates random output needs a random seed, e.g., an arbitrarily chosen integer, for initialization purposes. For instance, in recommender systems, when data should be randomly split, the sampling function requires a random seed to produce a random selection of elements. A random generator, e.g., the aforementioned sampling function, always produces the same chain of outputs when initialized with the same random seed. When the random seed is explicitly set, repeated executions always run with the same randomization, e.g., the same data split from the same input data. This is generally desirable and ensures the reproducibility and repeatability of recommender systems research. However, the downside is that repeated experiments with the same random seed remove randomness from the evaluation, ignoring its potentially severe effects. We illustrate this with a recommender systems example in the following paragraph.

When recommender systems are evaluated with a randomized holdout split, the data is split with one specific random seed. If experiments are not repeated with different holdout splits, e.g., different random seeds, the evaluation of the recommender systems algorithm is not protected against the impact of the randomness of data on the evaluated recommendation performance. To illustrate, each data split, generated through a specific random seed, may be different in its data distribution, e.g., the interactions that make up the training and the testing set. Specifically in recommender systems, some users or items may evaluate with exceptionally low recommendation accuracy while others do the opposite. Ideally, the data should be split in a way that results in an average over these outliers, but this can not be guaranteed. In the worst case, the evaluated accuracy of an algorithm on one specific split could be an outlier that significantly changes the judgment of a specific parameter setup or algorithm. This can also be abused to obtain unnaturally good results that are not achievable on average. Furthermore, replicating the exact result is unlikely if the random seed used in an experiment is unknown. We assume that the effect of the randomness of data on the evaluated performance of recommender systems algorithms is significant enough to warrant methods that mitigate the shortcomings of a single holdout split.

Paper Study To motivate a thorough analysis of our suspicion, we conducted a brief ad-hoc study of the 39 full papers that were published at the *ACM RecSys Conference 2022*[1]. We analyzed (1) which type of validation procedure was used in the experiments, (2) whether the code and random seeds used for the experiments are public, and (3) whether the authors acknowledged randomness and repeated experiments that are influenced by randomness. Regarding (1), we found that 26 papers (66.6%) use some form of holdout split to validate their results. Of the remaining papers, 10 (25.6%) used time-based leave-one-out splits that do not require randomness, and the remaining three (7.7%) did not perform and evaluate any experiments. Regarding (2), we found that 20 papers (51.3%) contain links to code to reproduce the results found in the papers. Of those 20 papers, 11 (28.2%) allow for the configuration of random seeds and provide default values, 6 (15.4%) contain static random seeds, and three (7.7%) do not

[1]https://recsys.acm.org/recsys22/accepted-contributions/

specify a random seed at all. Reproducing the results of papers without the original code is difficult and time-consuming. The task is even more challenging since no authors explicitly state which random seeds were used. However, even in the papers that contain links to code, it is unclear which random seed was used to obtain the results presented in the paper. Regarding (3), we found that only five (12.8%) papers acknowledge and deal with the effects of randomness during data-splitting on the recommendation accuracy. The authors of these papers found that repeating their experiments three, five, or ten times with different random seeds was an appropriate solution.

Research Question We find the current practices surrounding random seeds, especially during the data-splitting phase, worrying. Given that so many papers apply holdout splits but do not repeat their experiments, we seek to answer the following research question: **How significant is the impact of random seeds used for splitting the data on recommendation accuracy?**

We answer the research question by evaluating nine different data sets on three different recommendation algorithms with 20 different random seeds. We provide results for the *nDCG@k* and *Precision@k* metrics with $k = \{1, 5, 10\}$. Furthermore, we propose cross-validation, a standard practice in machine learning, as a mitigation strategy, and we showcase how it compares to the results obtained with a holdout split. Our contribution quantifies the effect of random seeds used during the data-splitting phase on recommendation accuracy. With this analysis, we aim to increase the awareness of the recommender systems community of the importance of random seeds for evaluating recommender systems. Furthermore, the code is open-source[2], and our experiments are reproducible.

2. Related Work

In machine learning and deep learning, the effects of random seeds are well-explored. The holdout split is a traditional method to validate machine learning models. Early works suggest that repeated holdout, with different random seeds, is necessary to correctly estimate the error of a given configuration [5]. Furthermore, in machine learning literature, statistical tests over multiple algorithms and data sets are only reliable when experiments are repeated sufficiently often [6]. In another article, the authors describe how static random seeds can impact the validation procedure [7]. Specifically in deep learning, multiple authors report how random seeds in model initialization affect performance [8, 9]. Further evidence is that this also extends to the data split [10]. Moreover, some works show the strong effects of random seeds on the stability of deep learning models [11, 12]. This effect is specifically exploited to improve deep learning ensemble performance [13, 14]. Apart from misrepresented performance, the reported statistical significance in comparing two algorithms may also be misleading due to poor random seeding practices [15]. Due to its advantages, cross-validation has become the standard for machine learning a long time ago [16]. The AutoML community understands the importance of randomness, and the collaborative AutoML benchmark uses 10-fold cross-validation to mitigate randomness by default [17, 18].

[2]https://code.isg.beel.org/random-seed-effects/

Recommender system evaluation is notoriously complex with many domain-specific problems but also inherits the general machine learning problems stated above [19, 20]. To our knowledge, there is no survey about using validation strategies in recommender systems research. In our paper study (⟶Introduction), we found that holdout validation is still a widespread data-splitting method. However, only a few authors apply repeated holdout validation, citing the effects of randomness [21, 22, 23, 24, 25]. Additionally, while acknowledging randomness, none of the authors cite any reference concerning randomness. We are unaware of any work that concretely analyzes and quantifies this effect.

Performing random holdout splits neglects temporal effects. In our paper study (⟶Introduction) we found that a significant amount of researchers use time-based splitting techniques. However, the majority still applies a random holdout split. Recommender systems are inherently affected by the chronological order and distance of historical interactions, and ignoring this when splitting the data may lead to temporal leakage [26, 27]. Our work does not focus on temporal effects.

Overall, there are only a few works in the recommender systems literature that understand the importance of and use cross-validation [28, 29, 30, 31]. Interestingly, many popular recommender systems libraries also natively include cross-validation [32, 33, 34, 35]. However, in our paper study (⟶Introduction), we found that no authors use cross-validation. We are unaware of any work that analyzes and quantifies how cross-validation mitigates the effects of the randomness of data compared to holdout, specifically for recommender systems.

3. Method

Our experiments showcase the effects of random seeds used during the data-splitting phase on recommendation accuracy. We denote such a random seed as *data split random seed* from here on to distinguish it from usages of random seeds in other evaluation components. We evaluate our pipeline with common design choices found in the literature such that the effects we show in our results apply to many research results. Therefore, the analysis focuses on the top-n ranking prediction task where the input data consists of binary user-item interactions.

Data Sets & Algorithms We evaluate nine publicly available and commonly used data sets: *Adressa [36]*, *Amazon-CDs&Vinyl [37]*, *Gowalla [38]*, *Hetrec-LastFM [39]*, *MovieLens-1M [40]*, *Amazon-MusicalInstruments [37]*, *Retailrocket*[3], *Amazon-VideoGames [37]*, and *Yelp*[4]. Table 1 shows statistical information on the data sets. We evaluate the data sets on three algorithms from traditionally used categories: *Implicit Matrix Factorization with Alternating Least Squares (ALS)*, *Item-based k-Nearest Neighbors (ItemKNN)*, and the baseline recommender *Popularity Recommender (Pop)*. We use the algorithm implementations from the LensKit library [41].

Preprocessing For the explicit feedback data sets *Amazon* and *MovieLens*, we treat a rating of higher than three as an interaction according to common practice [42, 43, 44]. For all data sets, we remove duplicates and incomplete entries, and all features other than the user ID and

[3]https://www.kaggle.com/datasets/retailrocket/ecommerce-dataset
[4]https://www.yelp.com/dataset

item ID. Following previous works, we perform 5-core filtering, which means that we prune the data to ensure that all users and items contain at least five interactions [45, 46, 47].

Table 1
Basic information of the data sets used in our experiments.

	#Interactions	#Users	#Items	Avg.#Int. per user	Avg.#Int. per item	Sparsity
adressa	2,020,328	146,635	2,441	13.78	827.66	99.44%
cds-and-vinyl	1,075,615	87,712	59,934	12.26	17.95	99.98%
gowalla	2,018,421	64,115	164,532	31.48	12.27	99.98%
hetrec-lastfm	71,355	1,859	2,823	38.38	25.28	98.64%
movielens-1m	574,376	6,034	3,125	95.19	183.8	96.95%
musical-instruments	147,173	18,177	7,414	8.1	19.85	99.89%
retailrocket	240,938	22,178	17,803	10.86	13.53	99.94%
video-games	291,985	33,625	12,455	8.68	23.44	99.93%
yelp	3,999,684	268,658	109,340	14.89	36.58	99.99%

Data Splitting Our primary analysis revolves around the effect of different *data split random seeds* on recommendation accuracy. We want to show how different *data split random seeds* affect the common practice of holdout splits and compare this to the machine learning standard of cross-validation. Therefore, we perform 20 holdout splits with different *data split random seeds* for each data set. We split the interactions with a ratio of 80% for training and 20% for testing, following other works [48, 49, 50]. For cross-validation, we use the same *data split random seeds* to generate five non-overlapping testing folds for each data set 20 times.

Training & Evaluation We train a model for each recommendation algorithm per data set per *data split random seed*. For each *data split random seed*, we train one model with the holdout training data and evaluate it on the holdout testing data. Additionally, for cross-validation, we train five separate models with the training data of all five data split folds, evaluate them on the respective testing data, and average the results. We train all models with default algorithm hyperparameters as we look for the effects of *data split random seeds* without the confounding effects of hyperparameter optimization. We then evaluate the predictions with the *nDCG@k* and *Precision@k* metrics with $k = \{1, 5, 10\}$. We choose different values for k to analyze the effects at various cutoff points.

4. Results & Conclusion

To answer our research question, we quantify the expected variation in recommendation accuracy of different *data split random seeds* when using holdout validation and compare this to cross-validation. Therefore, we present the results of our experiments over all nine data sets separated by the three algorithms, two metrics, three different cutoff values, and two validation methods.

Figure 1 shows a comprehensive summary of these experiments. There is one plot per algorithm, per metric with each cutoff value k, and each plot compares holdout to cross-validation on the y-axis. On the x-axis, the value 100 designates the mean accuracy of the evaluations of one data set with 20 different *data split random seeds*. Furthermore, each plot contains the evaluations of all nine data sets. To illustrate, we observe a few outliers for both validation methods by looking at the plot for the algorithm *ALS* evaluated on *Precision@1*. One of them is at ~106 for holdout, which means that there exists a *data split random seed* that achieved an accuracy that is ~6% higher than the accuracy for its data set on average. Conversely, another achieved a ~5.5% lower accuracy than the mean for its data. Finally, we used a Wilcoxon test ($\alpha = 0.05$) to verify whether the hypothesis H_0, that the distributions of the absolute deviation from the mean across seeds are equal for holdout and cross-validation, holds per algorithm. We were able to reject H_0 for each algorithm with $p < 0.001$. Therefore, the impact of the *data split random seed* on the evaluation of recommendation accuracy is significantly different for holdout and cross-validation for all tested algorithms.

In the following paragraphs, we point to the main observations of these results, interpret their meaning for our research goal, outline the limitations of this work, and conclude by answering our research questions.

Observations We highlight seven observations that are numbered for reference: (1) In the most extreme case of the non-baseline recommenders, which is *ItemKNN* evaluated with *Precision@1*, there is one *data split random seed* that resulted in ~6.3% higher accuracy than the mean with a holdout split. (2) In the same case, since there is also a *data split random seed* that achieves a ~5.3% lower recommendation accuracy, the best-performing *data split random seed* has a ~12.2% higher accuracy than worst-performing *data split random seed.* (3) Cross-validation makes the effect much less noticeable, where the highest deviation is only ~2.3% over the mean. (4) Similarly, the lowest-performing *data split random seed* for cross-validation is only ~1.8% lower than the mean, resulting in an accuracy range of ~4.2%. (5) In terms of the effect of *data split random seeds* on accuracy, *ALS* and *ItemKNN* are similar across the board, while *Pop* has a larger range and more outliers. (6) We observe that the deviation shrinks with bigger *k*, but it does so proportionally for both holdout and cross-validation. (7) There is no noticeable difference in the observations between *nDCG* and *Precision*, and we note that *nDCG@1* and *Precision@1* are equivalent.

Interpretation Observation (1) shows that specific *data split random seeds* can result in extreme performance gain over the mean performance. Therefore, it is possible to cherry-pick a *data split random seed* that significantly overestimates the performance of an algorithm. Similarly, observation (2) shows that the opposite may be the effect. Results that are outliers in either direction can be achieved if the *data split random seed* is chosen negligently, potentially skewing results. Specifically, a static *data split random seed* may return extreme results, which can go unnoticed in repeated experiments. Observations (3) and (4) show how cross-validation mitigates this effect by drastically reducing the magnitude of outliers. Furthermore, a logical conclusion is that if cross-validation can not be applied for any reason, averaging results over repeated holdout with different *data split random seeds* is still expected to result in an accuracy closer to the mean. Observation (5) shows how much the common baseline algorithm *Pop* reacts to the randomness of data and how the traditional algorithms *ALS* and *ItemKNN* are similarly affected by the randomness of data. Observation (6) may be explained by the fact that larger

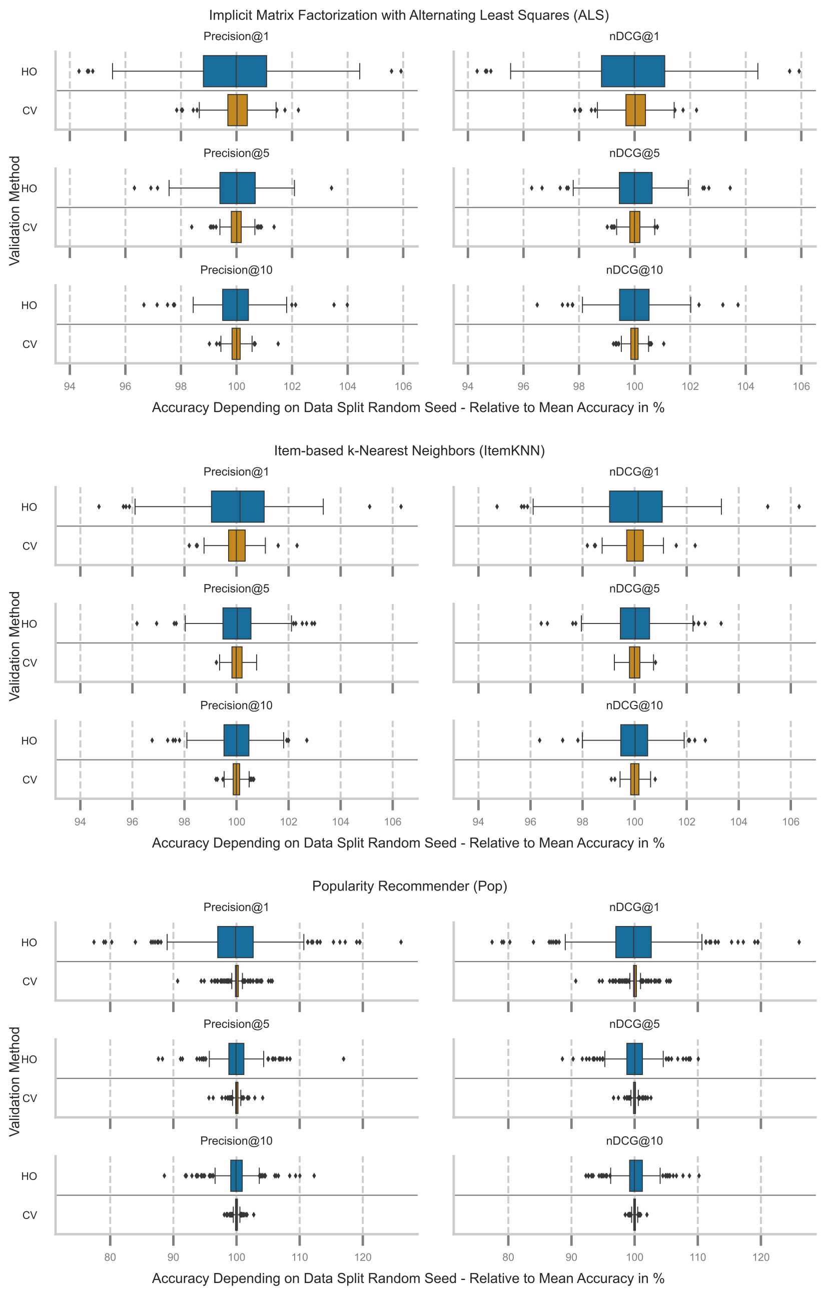

Figure 1: Results of the entire suite of experiments over all nine data sets and three algorithms. The plots are separated by algorithm, metric, and cutoff value *k*. The validation methods on the y-axis of each plot are *Holdout Validation (HO)* and *Five-Fold Cross-Validation (CV)*. The x-axis shows the relative performance deviation of *data split random seeds* in recommendation accuracy from the mean accuracy over all 20 *data split random seeds*. Each plot further aggregates all nine data sets, so there are 180 data points in total, with 20 *data split random seeds* for each of the nine data sets.

cutoff values *k* lead to a higher probability that the recommended items contain ground-truth interactions, reducing the impact of *data split random seeds* on the accuracy. However, while the absolute deviation shrinks with larger *k*, cross-validation still mitigates the impact of *data split random seeds* by a similar magnitude. Observation (7) is a hint to the similarity of *Precision* and *nDCG* in recommender systems and shows that *data split random seeds* similarly affect both metrics that do and do not account for the position of the recommended items.

Limitations The results are obtained with non-optimized models. We acknowledge that hyperparameter optimization may have an effect on the reported distribution. It may be worthwhile to repeat the experiments with hyperparameter optimization in the future but that is an analysis with a different objective and incurs relatively high computational requirements. Furthermore, we did not analyze the effects of *data split random seeds* with respect to data set metadata. There may be a connection between the data set size, sparsity, and other features and the variance of resulting performance scores. However, such a detailed analysis is outside the scope of this paper. Still, it may be interesting to understand how much data set characteristics affect the severity of the effects of *data split random seeds* in future work.

Conclusion We answer our research question by quantifying **how significant the impact of *data split random seeds* on recommendation accuracy is**. Our results, observations, and interpretation show that the impact of *data split random seeds* on recommendation accuracy may be significant to warrant steps to mitigate it. Even with a cutoff value of 10, e.g., *ALS* evaluated with *Precision@10*, any one *data split random seed* may still lead to accuracy that is up to ~4% lower than the mean accuracy over all *data split random seeds*, potentially changing the ranks of compared approaches. To illustrate, if an experiment evaluates an algorithm on *Precision@10* without repetition on a *data split random seed* that produces an exceptionally low accuracy, it may be assumed that it is up to ~4% worse than it would be on average. On the contrary, if a new algorithm is evaluated with the same parameters, it could be estimated to be ~4% better than on average, potentially opening a rift between the two compared algorithms.

Given these results, we argue it is hard to trust a result evaluated with holdout validation on a single *data split random seed.* We urge researchers to take randomness into account when evaluating an algorithm properly. Due to our results, we are convinced that a single holdout evaluation is not enough to gauge an evaluation pipeline's performance accurately. Given the distributions shown in Figure 1 and that we were able to reject H_0 with $p < 0.001$ for each algorithm, we recommend performing at least repeated holdout validation or cross-validation, and in the best case, repeated cross-validation. We acknowledge that experimenting may be expensive, but misjudging algorithm performance may be more costly in the long term.

Acknowledgements

The OMNI cluster of the University of Siegen was used to compute the results presented in this paper.

References

[1] E. Zangerle, C. Bauer, Evaluating recommender systems: Survey and framework, ACM Comput. Surv. 55 (2022). URL: https://doi.org/10.1145/3556536. doi:10.1145/3556536.

[2] W. Cai, W. Pan, J. Mao, Z. Yu, C. Xu, Aspect re-distribution for learning better item embeddings in sequential recommendation, in: Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 49–58. URL: https://doi.org/10.1145/3523227.3546764. doi:10.1145/3523227.3546764.

[3] W. Song, S. Wang, Y. Wang, S. Wang, Next-item recommendations in short sessions, in: Proceedings of the 15th ACM Conference on Recommender Systems, RecSys '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 282–291. URL: https://doi.org/10.1145/3460231.3474238. doi:10.1145/3460231.3474238.

[4] R. Ma, N. Liu, J. Yuan, H. Yang, J. Zhang, Caen: A hierarchically attentive evolution network for item-attribute-change-aware recommendation in the growing e-commerce environment, in: Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 278–287. URL: https://doi.org/10.1145/3523227.3546773. doi:10.1145/3523227.3546773.

[5] J.-H. Kim, Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap, Computational Statistics & Data Analysis 53 (2009) 3735–3745. URL: https://www.sciencedirect.com/science/article/pii/S0167947309001601. doi:https://doi.org/10.1016/j.csda.2009.04.009.

[6] J. Demšar, Statistical comparisons of classifiers over multiple data sets, The Journal of Machine learning research 7 (2006) 1–30.

[7] A. D'Amour, K. Heller, D. Moldovan, B. Adlam, B. Alipanahi, A. Beutel, C. Chen, J. Deaton, J. Eisenstein, M. D. Hoffman, F. Hormozdiari, N. Houlsby, S. Hou, G. Jerfel, A. Karthikesalingam, M. Lucic, Y. Ma, C. McLean, D. Mincu, A. Mitani, A. Montanari, Z. Nado, V. Natarajan, C. Nielson, T. F. Osborne, R. Raman, K. Ramasamy, R. Sayres, J. Schrouff, M. Seneviratne, S. Sequeira, H. Suresh, V. Veitch, M. Vladymyrov, X. Wang, K. Webster, S. Yadlowsky, T. Yun, X. Zhai, D. Sculley, Underspecification presents challenges for credibility in modern machine learning, 2020. arXiv:2011.03395.

[8] J. Phang, T. Févry, S. R. Bowman, Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks, 2019. arXiv:1811.01088.

[9] S. Amir, J.-W. van de Meent, B. C. Wallace, On the impact of random seeds on the fairness of clinical classifiers, 2021. arXiv:2104.06338.

[10] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, N. Smith, Fine-tuning pre-trained language models: Weight initializations, data orders, and early stopping, 2020. arXiv:2002.06305.

[11] P. Madhyastha, R. Jain, On model stability as a function of random seed, 2019. arXiv:1909.10447.

[12] M. Mosbach, M. Andriushchenko, D. Klakow, On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines, 2021. arXiv:2006.04884.

[13] Z.-H. Zhou, J. Wu, W. Tang, Ensembling neural networks: Many could be better than all, Artificial Intelligence 137 (2002) 239–263. URL: https://www.sciencedirect.com/science/

article/pii/S000437020200190X. doi:https://doi.org/10.1016/S0004-3702(02)00190-X.

[14] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, 2017. arXiv:1612.01474.

[15] C. Colas, O. Sigaud, P.-Y. Oudeyer, How many random seeds? statistical power analysis in deep reinforcement learning experiments, 2018. arXiv:1806.08295.

[16] S. Arlot, A. Celisse, A survey of cross-validation procedures for model selection, Statistics Surveys 4 (2010) 40 – 79. URL: https://doi.org/10.1214/09-SS054. doi:10.1214/09-SS054.

[17] P. Gijsbers, E. LeDell, J. Thomas, S. Poirier, B. Bischl, J. Vanschoren, An open source automl benchmark, 2019. arXiv:1907.00909.

[18] P. Gijsbers, M. L. P. Bueno, S. Coors, E. LeDell, S. Poirier, J. Thomas, B. Bischl, J. Vanschoren, Amlb: an automl benchmark, 2022. arXiv:2207.12560.

[19] F. Hernández del Olmo, E. Gaudioso, Evaluation of recommender systems: A new approach, Expert Systems with Applications 35 (2008) 790–804. URL: https://www.sciencedirect.com/science/article/pii/S0957417407002928. doi:https://doi.org/10.1016/j.eswa.2007.07.047.

[20] A. Said, A. Bellogín, Comparative recommender system evaluation: Benchmarking recommendation frameworks, in: Proceedings of the 8th ACM Conference on Recommender Systems, RecSys '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 129–136. URL: https://doi.org/10.1145/2645710.2645746. doi:10.1145/2645710.2645746.

[21] Z. He, H. Zhao, T. Yu, S. Kim, F. Du, J. McAuley, Bundle mcr: Towards conversational bundle recommendation, in: Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 288–298. URL: https://doi.org/10.1145/3523227.3546755. doi:10.1145/3523227.3546755.

[22] S. Borg Bruun, M. Maistro, C. Lioma, Learning recommendations from user actions in the item-poor insurance domain, in: Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 113–123. URL: https://doi.org/10.1145/3523227.3546775. doi:10.1145/3523227.3546775.

[23] H. Chen, X. Li, K. Zhou, X. Hu, C.-C. M. Yeh, Y. Zheng, H. Yang, Tinykg: Memory-efficient training framework for knowledge graph neural recommender systems, in: Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 257–267. URL: https://doi.org/10.1145/3523227.3546760. doi:10.1145/3523227.3546760.

[24] C. Almagor, Y. Hoshen, You say factorization machine, i say neural network - it's all in the activation, in: Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 389–398. URL: https://doi.org/10.1145/3523227.3551499. doi:10.1145/3523227.3551499.

[25] A. Rashed, S. Elsayed, L. Schmidt-Thieme, Context and attribute-aware sequential recommendation via cross-attention, in: Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 71–80. URL: https://doi.org/10.1145/3523227.3546777. doi:10.1145/3523227.3546777.

[26] O. Jeunen, Revisiting offline evaluation for implicit-feedback recommender systems, in: Proceedings of the 13th ACM Conference on Recommender Systems, RecSys '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 596–600. URL: https:

//doi.org/10.1145/3298689.3347069. doi:10.1145/3298689.3347069.

[27] Y. Ji, A. Sun, J. Zhang, C. Li, A critical study on data leakage in recommender system offline evaluation, ACM Trans. Inf. Syst. 41 (2023). URL: https://doi.org/10.1145/3569930. doi:10.1145/3569930.

[28] L. Brozovsky, V. Petricek, Recommender system for online dating service, 2007. arXiv:cs/0703042.

[29] A. Košir, A. Odić, M. Tkalčič, How to improve the statistical power of the 10-fold cross validation scheme in recommender systems, in: Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation, RepSys '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 3–6. URL: https://doi.org/10.1145/2532508.2532510. doi:10.1145/2532508.2532510.

[30] E. A. Sosnina, S. Sosnin, A. A. Nikitina, I. Nazarov, D. I. Osolodkin, M. V. Fedorov, Recommender systems in antiviral drug discovery, ACS Omega 5 (2020) 15039–15051. URL: https://doi.org/10.1021/acsomega.0c00857. doi:10.1021/acsomega.0c00857. arXiv:https://doi.org/10.1021/acsomega.0c00857, pMID: 32632398.

[31] D. I. Ignatov, J. Poelmans, G. Dedene, S. Viaene, A new cross-validation technique to evaluate quality of recommender systems, in: M. K. Kundu, S. Mitra, D. Mazumdar, S. K. Pal (Eds.), Perception and Machine Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 195–202.

[32] M. D. Ekstrand, Lenskit for python: Next-generation software for recommender systems experiments, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 2999–3006. URL: https://doi.org/10.1145/3340531.3412778. doi:10.1145/3340531.3412778.

[33] N. Hug, Surprise: A python library for recommender systems, Journal of Open Source Software 5 (2020) 2174.

[34] V. W. Anelli, A. Bellogin, A. Ferrara, D. Malitesta, F. A. Merra, C. Pomo, F. M. Donini, T. Di Noia, Elliot: A comprehensive and rigorous framework for reproducible recommender systems evaluation, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 2405–2414. URL: https://doi.org/10.1145/3404835.3463245. doi:10.1145/3404835.3463245.

[35] Z. Gantner, S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Mymedialite: A free recommender system library, in: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 305–308. URL: https://doi.org/10.1145/2043932.2043989. doi:10.1145/2043932.2043989.

[36] J. A. Gulla, L. Zhang, P. Liu, O. Özgöbek, X. Su, The adressa dataset for news recommendation, in: Proceedings of the International Conference on Web Intelligence, WI '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 1042–1048. URL: https://doi.org/10.1145/3106426.3109436. doi:10.1145/3106426.3109436.

[37] J. Ni, J. Li, J. McAuley, Justifying recommendations using distantly-labeled reviews and fine-grained aspects, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong

Kong, China, 2019, pp. 188–197. URL: https://aclanthology.org/D19-1018. doi:10.18653/v1/D19-1018.

[38] E. Cho, S. A. Myers, J. Leskovec, Friendship and mobility: User movement in location-based social networks, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 1082–1090. URL: https://doi.org/10.1145/2020408.2020579. doi:10.1145/2020408.2020579.

[39] I. Cantador, P. Brusilovsky, T. Kuflik, 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011), in: Proceedings of the 5th ACM conference on Recommender systems, RecSys 2011, ACM, New York, NY, USA, 2011.

[40] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, ACM Trans. Interact. Intell. Syst. 5 (2015). URL: https://doi.org/10.1145/2827872. doi:10.1145/2827872.

[41] M. D. Ekstrand, M. Ludwig, J. A. Konstan, J. T. Riedl, Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit, in: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 133–140. URL: https://doi.org/10.1145/2043932.2043958. doi:10.1145/2043932.2043958.

[42] O. Barkan, R. Hirsch, O. Katz, A. Caciularu, N. Koenigstein, Anchor-based collaborative filtering, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 2877–2881. URL: https://doi.org/10.1145/3459637.3482056. doi:10.1145/3459637.3482056.

[43] A. B. Melchiorre, N. Rekabsaz, C. Ganhör, M. Schedl, Protomf: Prototype-based matrix factorization for effective and explainable recommendations, in: Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 246–256. URL: https://doi.org/10.1145/3523227.3546756. doi:10.1145/3523227.3546756.

[44] D. Liang, R. G. Krishnan, M. D. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: Proceedings of the 2018 World Wide Web Conference, WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018, p. 689–698. URL: https://doi.org/10.1145/3178876.3186150. doi:10.1145/3178876.3186150.

[45] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, P. Jiang, Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1441–1450. URL: https://doi.org/10.1145/3357384.3357895. doi:10.1145/3357384.3357895.

[46] Z. Yue, Z. He, H. Zeng, J. McAuley, Black-box attacks on sequential recommenders via data-free model extraction, in: Proceedings of the 15th ACM Conference on Recommender Systems, RecSys '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 44–54. URL: https://doi.org/10.1145/3460231.3474275. doi:10.1145/3460231.3474275.

[47] Z. Yue, H. Zeng, Z. Kou, L. Shang, D. Wang, Defending substitution-based profile pollution attacks on sequential recommenders, in: Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, Association for Computing Machinery, New York, NY,

USA, 2022, p. 59–70. URL: https://doi.org/10.1145/3523227.3546770. doi:10.1145/3523227.3546770.

[48] X. Wang, X. He, Y. Cao, M. Liu, T.-S. Chua, Kgat: Knowledge graph attention network for recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 950–958. URL: https://doi.org/10.1145/3292500.3330989. doi:10.1145/3292500.3330989.

[49] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Analysis of recommendation algorithms for e-commerce, in: Proceedings of the 2nd ACM Conference on Electronic Commerce, EC '00, Association for Computing Machinery, New York, NY, USA, 2000, p. 158–167. URL: https://doi.org/10.1145/352871.352887. doi:10.1145/352871.352887.

[50] T. Avny Brosh, A. Livne, O. Sar Shalom, B. Shapira, M. Last, Bruce: Bundle recommendation using contextualized item embeddings, in: Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 237–245. URL: https://doi.org/10.1145/3523227.3546754. doi:10.1145/3523227.3546754.