Introducing LensKit-Auto, an Experimental Automated Recommender System (AutoRecSys) Toolkit

Tobias Vente
tobias.vente@uni-siegen.de
University of Siegen, Intelligent Systems Group
Siegen, Germany

Michael D. Ekstrand
michaelekstrand@boisestate.edu
Boise State University
Boise, ID, United States of America

Joeran Beel
joeran.beel@uni-siegen.de
University of Siegen, Intelligent Systems Group
Siegen, Germany

ABSTRACT

LensKit is one of the first and most popular Recommender System libraries. While LensKit offers a wide variety of features, it does not include any optimization strategies or guidelines on how to select and tune LensKit algorithms. LensKit developers have to manually include third-party libraries into their experimental setup or implement optimization strategies by hand to optimize hyperparameters. We found that 63.6% (21 out of 33) of papers using LensKit algorithms for their experiments did not select algorithms or tune hyperparameters. Non-optimized models represent poor baselines and produce less meaningful research results. This demo introduces LensKit-Auto. LensKit-Auto automates the entire Recommender System pipeline and enables LensKit developers to automatically select, optimize, and ensemble LensKit algorithms.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender Systems, Automated Recommender Systems, *AutoRecSys*, Algorithm Selection, Hyperparameter Optimization, CASH

ACM Reference Format:
Tobias Vente, Michael D. Ekstrand, and Joeran Beel. 2023. Introducing LensKit-Auto, an Experimental Automated Recommender System (AutoRecSys) Toolkit. In *Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, Singapore.* ACM, New York, NY, USA, Article 111, 5 pages. https://doi.org/10.1145/3604915.3610656

1 INTRODUCTION

LensKit [18] is one of the first Recommender Systems libraries, published in 2011 as a Java library [20], and more recently republished as a Python library [18]. In the last five years, it was widely used for Recommender Systems research presented at premier venues such as ACM RecSys [22], SIGIR [41, 55, 55], CIKM [16, 19] and various journals [4–6, 14, 28, 38, 40, 46, 51, 58], workshops [57], conferences [7, 11, 12, 17, 21, 48, 52, 62] and other projects [9, 32, 42, 43, 47, 54, 65].

RecSys '23, September 18–22, 2023, Singapore, Singapore

ACM ISBN 979-8-4007-0241-9/23/09.
https://doi.org/10.1145/3604915.3610656

While LensKit has many features that allow Recommender Systems developers and researchers to easily prototype Recommender Systems and conduct research, LensKit is missing one important feature. LensKit offers no built-in hyperparameter optimization methods. Consequently, users of LensKit need to manually optimize hyperparameters and select algorithms with additional tools like scikit-optimize [31] or Elliot [3]. Alternatively, they do not perform algorithm selection and hyperparameter optimization at all. In a survey of 33 papers [4–7, 9, 11, 12, 14, 16, 17, 19, 21, 22, 28, 38–43, 46–48, 51, 52, 54, 55, 55, 57, 58, 62, 65, 66] from the past five years that used LensKit for their experiments, only 36.4% of papers optimized hyperparameters with external tools. The remaining 63.6% (21 out of 33) did not optimize hyperparameters or did not mention it. This is a substantial problem because non-optimized algorithms represent poor baselines and produce less meaningful research results [8, 25, 36].

In comparison to the field of Recommender Systems, the field of Machine Learning offers quick and easy solutions to perform algorithm selection and hyperparameter optimization [56]. In Machine Learning, an entirely new research field evolved, solely dedicated to the automation of, e.g., algorithm selection and hyperparameter optimization: Automated Machine Learning (AutoML) [25, 34]. AutoML methods go beyond standard grid or random search but cover the entire Machine Learning pipeline, including pre- and postprocessing. AutoML also includes enhanced methods like Bayesian optimization [59] and post-hoc ensembling. Post-hoc ensembling offers an additional boost in the performance over algorithm selection and hyperparameter optimization [24, 53]. In post-hoc ensembling, the top-performing models of the optimization process are efficiently re-used and combined in an ensemble [10]. 55% (5 out of 9) of AutoML libraries evaluated in the latest AutoML Benchmark [27] use post-hoc ensembling by default. A Machine Learning developer quickly executes AutoML methods with a single function call by using one of the multiple AutoML libraries like Auto-sklearn [26], H2O [35], AutoGluon [23], FLAML [63], TPOT [49], or Auto-PyTorch [68]. As a result, a developer achieves state-of-the-art performance with AutoML tools without spending time selecting algorithms or tuning hyperparameters [34].

The success of AutoML inspired the Recommender Systems community and led to the first development of Automated Recommender Systems libraries (*AutoRecSys*) in recent years. *AutoRecSys* libraries often extend existing "normal" Recommender System libraries. For instance, LibRec-Auto [60] extends the LibRec library [29]; AutoSurprise [2] extends the Surprise library [33]; Auto-CaseRec [30] extends the CaseRec library [13]; AutoRec [64] extends Tensorflow Recommenders [1] and BETA-Rec [45] extends

Table 1: LensKit-Auto compared to existing *AutoRecSys* tools

	Default Configuration Space	Automated Preprocessing	Algorithm Selection	Black-Box Optimization	Automated Postprocessing
LibRec-Auto	-	-	-	✓	(-)
AutoSurprise	✓	-	✓	✓	-
Auto-CaseRec	✓	-	✓	✓	-
AutoRec	-	-	-	✓	-
BetaRec	✓	-	-	✓	-
LensKit-Auto	✓	✓	✓	✓	✓

TorchRec [50]. In contrast to AutoML, *AutoRecSys* libraries do not yet offer advanced techniques like automated pre-processing and post-hoc ensembling.

In this demo, we introduce LensKit-Auto. LensKit-Auto is an *AutoRecSys* toolkit that extends LensKit with automatic algorithm selection, hyperparameter optimization and advanced AutoML methods like post-hoc ensembling. LensKit-Auto enables LensKit developers to perform algorithm selection, hyperparameter optimization and post-hoc ensembling for Top-N recommender and rating prediction tasks while minimizing human effort. In comparison to all other *AutoRecSys* Libraries (Table 1) LensKit-Auto automates the whole Recommender System pipeline and supports all three steps of algorithm selection, hyperparameter optimization and post-hoc ensembling. Furthermore, LensKit-Auto does not require LensKit developers to invest their time in manually creating configuration files or defining hyperparameter ranges.

2 LENSKIT-AUTO

LensKit-Auto[1] is a flexible Automated Recommender System (*AutoRecSys*) toolkit based on LensKit [18]. LensKit-Auto is open-source and performs automated algorithm selection, hyperparameter optimization, and post-hoc model ensembling on all algorithms included in the LensKit Python library for rating prediction and Top-N ranking datasets. LensKit-Auto strives to maintain a test coverage of more than 90% at all times. In comparison to other *AutoRecSys* tools, a Recommender Systems developer using LensKit-Auto does not need to spend time on defining configuration files or hyperparameter settings but only needs to execute a single line of code. LensKit-Auto then outputs the best-performing model for the specific dataset.

Furthermore, LensKit-Auto has three unique features that no other *AutoRecSys* tools offer.

- Bayesian optimization optimization with the state-of-the-art optimizer SMAC3 [37]
- Automatic post-hoc ensembling
- Automatic preprocessing steps like data-pruning

A typical experimentation pipeline to select a Recommender System model is shown in Figure 1. LensKit-Auto automates every step of the pipeline. The following section will discuss all steps and functionalities of the fully automated LensKit-Auto pipeline.

Data & Task: The Recommender System developer calls the *get_best_recommender_model()* function to get the best-performing model for Top-N recommendation tasks or the *get_best_prediction_model()* function to get the best-performing model for rating prediction tasks. Both functions take a *pandas DataFrame* [44] as the input. The *pandas DataFrame* is LensKit-Auto's only required input parameter. All other steps of the Recommender Systems pipeline can be configured to the Recommender Systems developer's needs with the parameters of the respective function call. LensKit-Auto's adjustable parameters are explained in the particular subsection.

Preprocessing: In the default preprocessing setting, the provided dataset remains unchanged. However, the Recommender Systems developer can utilize the automated preprocessing pipeline to remove duplicates and *None*-values interactions from the dataset. Furthermore, LensKit-Auto can prune users based on the minimum and maximum number of interactions.

Validation Split: The default validation split is a 75%:25% user-based holdout split [67]. LensKit-Auto also supports row-based splits. User-based and row-based validation splits can also be configured to be cross-fold splits. The number of folds depends on the Recommender Systems developer's choice.

Optimization: LensKit-Auto supports Bayesian optimization through SMAC3 [37] and random search [8] as optimization strategies. The stopping criterion of both optimization strategies is either time or the number of model evaluations. By default, LensKit-Auto searches for the best model using Bayesian optimization on all algorithms with one hour of search time per task. LensKit-Auto can also be configured to optimize the hyperparameters of a single LensKit algorithm or to search for the best model on a subset of LensKit algorithms. The LensKit-Auto library provides default hyperparameter search spaces for all of LensKit algorithms. The default hyperparameter ranges are based on the authors' advice from the original algorithm papers [15, 32, 61]. In the remaining cases, we evaluated the hyperparameter ranges in extensive experiments. For more details on our experiments, visit LensKit-Autos GitHub[1]. The developer easily adjusts the hyperparameter ranges for one or multiple algorithms. All LensKit metrics are supported as objective metrics. The default metric for the optimization process is *RMSE* for rating prediction tasks and *nDCG@10* for Top-N recommender tasks.

Postprocessing: After the optimization process, LensKit-Auto builds an ensemble using the Top-N evaluated models. By default, the top 50 models are ensembled. Following standard AutoML practices [24], LensKit-Auto implements the state-of-the-art Greedy

[1]https://github.com/ISG-Siegen/lenskit-auto

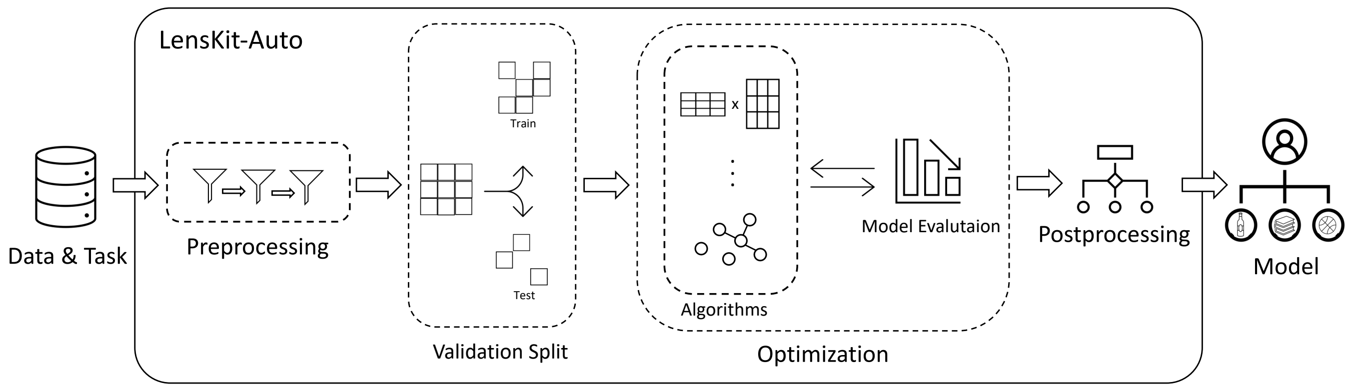

Figure 1: LensKit-Auto - Pipeline Diagram

Ensemble Selection method [10], currently only for rating prediction tasks. Greedy Ensemble Selection iteratively selects and then evaluates different model combinations on the validation data. In our implementation, Greedy Ensemble Selection is especially efficient because it reuses the predictions on the validation data produced and stored by LensKit-Auto. The final ensemble increases robustness, avoids overfitting and boosts performance compared to the single best model [24].

LensKit-Auto can be installed with pip or cloned from *GitHub*[1]. Detailed instructions on how to use LensKit-Auto can be found in the *Getting Started* chapter on GitHub. The code is open source.

3 DEMO

In the demo and our video recording on GitHub[1], we showcase the process of selecting the best-suited model in different scenarios. First, we showcase the process of integrating LensKit-Auto model selection into your recommendation experiment. We select the *Movielens100k* dataset and split it into a train and test set. Then, we pass the train set as a parameter to the LensKit-Auto *get_best_recommender_model()* function call. The function call returns the best-performing model for the *NDCG@10* metric. In the second part of the demo, we only tune the *BiasedMatrixFactorization* algorithm to perform well with respect to the *RMSE* metric using random search. We need to change the parameters of the *get_best_prediction_model()* function call to specifically select random search as an optimization strategy, change the optimization metric to *RMSE* and tune the *BiasedMatrixFactorization* algorithm.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Rohan Anand and Joeran Beel. 2020. Auto-Surprise: An Automated Recommender-System (AutoRecSys) Library with Tree of Parzens Estimator (TPE) Optimization. In *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*, Rodrygo L. T. Santos, Leandro Balby Marinho, Elizabeth M. Daly, Li Chen, Kim Falk, Noam Koenigstein, and Edleno Silva de Moura (Eds.). ACM, 585–587. https://doi.org/10.1145/3383313.3411467

[3] Vito Walter Anelli, Alejandro Bellogin, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. 2021. 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, 2405–2414. https://doi.org/10.1145/3404835.3463245

[4] Ashwathy Ashokan and Christian Haas. 2021. Fairness metrics and bias mitigation strategies for rating predictions. *Information Processing & Management* 58, 5 (Sept. 2021), 102646. https://doi.org/10.1016/j.ipm.2021.102646

[5] Michał Bałchanowski and Urszula Boryczka. 2022. Collaborative Rank Aggregation in Recommendation Systems. *Procedia computer science* 207 (Jan. 2022), 2213–2222. https://doi.org/10.1016/j.procs.2022.09.281

[6] Michał Bałchanowski and Urszula Boryczka. 2023. A Comparative Study of Rank Aggregation Methods in Recommendation Systems. *Entropy* 25, 1 (Jan. 2023), 132. https://doi.org/10.3390/e25010132

[7] Christine Bauer, Lennard Chung, Aleksej Cornelissen, Isabelle van Driessel, Diede van der Hoorn, Yme de Jong, Lan Le, Sanaz Najiyan Tabriz, Roderick Spaans, Casper Thijsen, Robert Verbeeten, Vos Wesseling, and Fern Wieland. 2023. FairRecKit: A Web-based Analysis Software for Recommender Evaluations. In *Proceedings of the 2023 Conference on Human Information Interaction and Retrieval* (Austin, TX, USA) *(CHIIR '23)*. Association for Computing Machinery, New York, NY, USA, 438–443. https://doi.org/10.1145/3576840.3578274

[8] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research* 13, 10 (2012), 281–305. http://jmlr.org/papers/v13/bergstra12a.html

[9] Parantapa Bhattacharya, Saptarshi Ghosh, Muhammad Bilal Zafar, Soumya K Ghosh, and Niloy Ganguly. 2022. What You Like: Generating Explainable Topical Recommendations for Twitter Using Social Annotations. (Dec. 2022). arXiv:2212.13897 [cs.IR] http://arxiv.org/abs/2212.13897

[10] Rich Caruana, Art Munson, and Alexandru Niculescu-Mizil. 2006. Getting the Most Out of Ensemble Selection. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 828–833. https://doi.org/10.1109/ICDM.2006.76

[11] Sushma Channamsetty and Michael D. Ekstrand. 2017. Recommender Response to Diversity and Popularity Bias in User Profiles. In *The Thirtieth International*

Flairs Conference. https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS17/paper/view/15524

[12] Ludovik Coba, Panagiotis Symeonidis, and Markus Zanker. 2018. Replicating and Improving Top-N Recommendations in Open Source Packages. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics (WIMS '18)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3227609.3227671

[13] Arthur da Costa, Eduardo Fressato, Fernando Neto, Marcelo Manzato, and Ricardo Campello. 2018. Case Recommender: A Flexible and Extensible Python Framework for Recommender Systems. In *Proceedings of the 12th ACM Conference on Recommender Systems* (Vancouver, British Columbia, Canada) *(RecSys '18)*. ACM, New York, NY, USA, 494–495. https://doi.org/10.1145/3240323.3241611

[14] Toon De Pessemier, Ine Coppens, and Luc Martens. 2020. Evaluating facial recognition services as interaction technique for recommender systems. *Multimedia Tools and Applications* 79, 31 (Aug. 2020), 23547–23570. https://doi.org/10.1007/s11042-020-09061-8

[15] Mukund Deshpande and George Karypis. 2004. Item-Based Top-N Recommendation Algorithms. *ACM Trans. Inf. Syst.* 22, 1 (jan 2004), 143–177. https://doi.org/10.1145/963770.963776

[16] Fernando Diaz, Bhaskar Mitra, Michael D. Ekstrand, Asia J. Biega, and Ben Carterette. 2020. Evaluating Stochastic Rankings with Expected Exposure. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 275–284. https://doi.org/10.1145/3340531.3411962

[17] Patrik Dokoupil and Ladislav Peska. 2023. The Effect of Similarity Metric and Group Size on Outlier Selection & Satisfaction in Group Recommender Systems. In *Adjunct Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization* (Limassol, Cyprus) *(UMAP '23 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 296–301. https://doi.org/10.1145/3563359.3597386

[18] Michael D. Ekstrand. 2020. LensKit for Python: Next-Generation Software for Recommender Systems Experiments. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management* (Virtual Event, Ireland) *(CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 2999–3006. https://doi.org/10.1145/3340531.3412778

[19] Michael D. Ekstrand. 2020. 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, 2999–3006. https://doi.org/10.1145/3340531.3412778

[20] Michael D. Ekstrand, Michael Ludwig, Joseph A. Konstan, and John T. Riedl. 2011. Rethinking the Recommender Research Ecosystem: Reproducibility, Openness, and LensKit. Association for Computing Machinery, New York, NY, USA.

[21] Michael D. Ekstrand, Mucun Tian, Ion Madrazo Azpiazu, Jennifer D. Ekstrand, Oghenemaro Anuyah, David McNeill, and Maria Soledad Pera. 2018. All The Cool Kids, How Do They Fit In?: Popularity and Demographic Biases in Recommender Evaluation and Effectiveness. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*. PMLR, 172–186. https://proceedings.mlr.press/v81/ekstrand18b.html ISSN: 2640-3498.

[22] Michael D. Ekstrand, Mucun Tian, Mohammed R. Imran Kazi, Hoda Mehrpouyan, and Daniel Kluver. 2018. Exploring author gender in book rating and recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18)*. Association for Computing Machinery, New York, NY, USA, 242–250. https://doi.org/10.1145/3240323.3240373

[23] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. AutoGluon-Tabular Robust and Accurate AutoML for Structured Data. *arXiv preprint arXiv2003.06505* (2020).

[24] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf

[25] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2* (Montreal, Canada) *(NIPS'15)*. MIT Press, Cambridge, MA, USA, 2755–2763.

[26] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (Eds.). 2962–2970. https://proceedings.neurips.cc/paper/2015/hash/11d0e6287202fced83f79975ec59a3a6-Abstract.html

[27] Pieter Gijsbers, Marcos L. P. Bueno, Stefan Coors, Erin LeDell, Sébastien Poirier, Janek Thomas, Bernd Bischl, and Joaquin Vanschoren. 2022. AMLB: an AutoML Benchmark. https://doi.org/10.48550/ARXIV.2207.12560

[28] Augustin Godinot and Fabien Tarissan. 2023. Measuring the effect of collaborative filtering on the diversity of users' attention. *Applied Network Science* 8, 1 (Jan. 2023). https://doi.org/10.1007/s41109-022-00530-7

[29] Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. 2015. LibRec: A Java Library for Recommender Systems.. In *UMAP Workshops (CEUR Workshop Proceedings, Vol. 1388)*, Alexandra I. Cristea, Judith Masthoff, Alan Said, and Nava Tintarev (Eds.). CEUR-WS.org. http://dblp.uni-trier.de/db/conf/um/umap2015w.html#GuoZSY15

[30] Srijan Gupta and Joeran Beel. 2020. Auto-CaseRec: Automatically Selecting and Optimizing Recommendation-Systems Algorithms. *OSF Preprints,* (2020). https://doi.org/10.31219/osf.io/4znmd

[31] Tim Head, Manoj Kumar, Holger Nahrstaedt, Gilles Louppe, and Iaroslav Shcherbatyi. 2021. scikit-optimize/scikit-optimize. https://doi.org/10.5281/zenodo.5565057

[32] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *2008 Eighth IEEE International Conference on Data Mining*. 263–272. https://doi.org/10.1109/ICDM.2008.22

[33] Nicolas Hug. 2020. Surprise A Python library for recommender systems. *Journal of Open Source Software* 5, 52 (2020), 2174. https://doi.org/10.21105joss.02174

[34] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren (Eds.). 2019. *Automated Machine Learning - Methods, Systems, Challenges*. Springer.

[35] Erin LeDell and Sebastien Poirier. 2020. H2O AutoML Scalable Automatic Machine Learning. *7th ICML Workshop on Automated Machine Learning (AutoML)* (2020). httpswww.automl.orgwp-contentuploads202007AutoML_2020_paper_61.pdf

[36] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. 2021. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. In *ArXiv: 2109.09831*. https://arxiv.org/abs/2109.09831

[37] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. 2022. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. *Journal of Machine Learning Research* 23, 54 (2022), 1–9. http://jmlr.org/papers/v23/21-0888.html

[38] Qin Liu. 2022. A New Collaborative Filtering Algorithm Integrating Time and Multisimilarity. *Mathematical Problems in Engineering* 2022 (Aug. 2022). https://doi.org/10.1155/2022/2340671

[39] Ramon Lopes, Rodrigo Alves, Antoine Ledent, Rodrygo Santos, and Marius Kloft. 2022. Recommendations with Minimum Exposure Guarantees: A Post-Processing Framework. (Nov. 2022). https://doi.org/10.2139/ssrn.4274780

[40] Hongyu Lu, Weizhi Ma, Yifan Wang, Min Zhang, Xiang Wang, Yiqun Liu, Tat-Seng Chua, and Shaoping Ma. 2022. User Perception of Recommendation Explanation: Are Your Explanations What Users Need? *ACM Transactions on Information and System Security* (Nov. 2022). https://doi.org/10.1145/3565480

[41] Hongyu Lu, Weizhi Ma, Min Zhang, Maarten de Rijke, Yiqun Liu, and Shaoping Ma. 2021. Standing in Your Shoes: External Assessments for Personalized Recommender Systems. In *Proc. SIGIR 2021*. https://doi.org/10.1145/3404835.3462916

[42] Eli Lucherini, Matthew Sun, Amy Winecoff, and Arvind Narayanan. 2021. T-RECS: A Simulation Tool to Study the Societal Impact of Recommender Systems. https://doi.org/10.48550/arXiv.2107.08959 arXiv:2107.08959 [cs].

[43] Yuhao Mao, Serguei A. Mokhov, and Sudhir P. Mudur. 2021. Application of Knowledge Graphs to Provide Side Information for Improved Recommendation Accuracy. https://doi.org/10.48550/arXiv.2101.03054 arXiv:2101.03054 [cs].

[44] Wes McKinney. 2010. Data Structures for Statistical Computing in Python. Austin, Texas, 56–61. https://doi.org/10.25080/Majora-92bf1922-00a

[45] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, Iadh Ounis, Siwei Liu, Yaxiong Wu, Xi Wang, Shangsong Liang, Yucheng Liang, Guangtao Zeng, Junhua Liang, and Qiang Zhang. 2020. *BETA-Rec: Build, Evaluate and Tune Automated Recommender Systems*. Association for Computing Machinery, New York, NY, USA, 588–590. https://doi.org/10.1145/3383313.3411524

[46] Lien Michiels, Robin Verachtert, Andres Ferraro, Kim Falk, and Bart Goethals. 2023. A Framework and Toolkit for Testing the Correctness of Recommendation Algorithms. *ACM Trans. Recomm. Syst.* (April 2023). https://doi.org/10.1145/3591109

[47] Anshuman Narayan. 2019. What is the Value of Rating Obscure Items? An Analysis of the Effect of Less-Popular Items on Recommendation Quality. (June 2019). http://conservancy.umn.edu/handle/11299/206148 Accepted: 2019-08-20T17:23:51Z.

[48] Yiu-Kai Ng. 2020. CBRec: a book recommendation system for children using the matrix factorisation and content-based filtering approaches. *International Journal of Business Intelligence and Data Mining* 16, 2 (Jan. 2020), 129–149. https://doi.org/10.1504/IJBIDM.2020.104738 Publisher: Inderscience Publishers.

[49] Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. 2016. Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016* (Denver, Colorado, USA) *(GECCO '16)*. ACM, New York, NY, USA, 485–492. https://doi.org/10.1145/2908812.2908918

[50] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban

Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. https://doi.org/10.48550/ARXIV.1912.01703

[51] Toon De Pessemier, Ine Coppens, and Luc Martens. 2019. Using facial recognition services as implicit feedback for recommenders. (2019), 8.

[52] Marco Polignano, Pierpaolo Basile, Marco de Gemmis, and Giovanni Semeraro. 2019. Social Tags and Emotions as main Features for the Next Song To Play in Automatic Playlist Continuation. In *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization (UMAP'19 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 235–239. https://doi.org/10.1145/3314183.3323455

[53] Lennart Oswald Purucker and Joeran Beel. 2022. Assembled-OpenML: Creating Efficient Benchmarks for Ensembles with OpenML. In *First Conference on Automated Machine Learning (Late-Breaking Workshop)*. https://openreview.net/forum?id=3UMmIzHVS3y

[54] Amifa Raj and Michael D. Ekstrand. 2022. Comparing Fair Ranking Metrics. https://doi.org/10.48550/arXiv.2009.01311 arXiv:2009.01311 [cs].

[55] Amifa Raj and Michael D Ekstrand. 2022. Measuring Fairness in Ranked Results: An Analytical and Empirical Comparison. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press. https://doi.org/10.1145/3477495.3532018

[56] Sebastian Raschka. 2018. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. *CoRR* abs/1811.12808 (2018). arXiv:1811.12808 http://arxiv.org/abs/1811.12808

[57] Teresa Scheidt and Joeran Beel. 2021. Time-dependent evaluation of recommender systems: 2021 Perspectives on the Evaluation of Recommender Systems Workshop, Perspectives 2021. *Perspectives 2021* 2955 (2021). http://www.scopus.com/inward/record.url?scp=85116263192&partnerID=8YFLogxK

[58] Manel Slokom, Alan Hanjalic, and Martha Larson. 2021. Towards user-oriented privacy for recommender system data: A personalization-based approach to gender obfuscation for user profiles. *Information Processing & Management* 58, 6 (Nov. 2021), 102722. https://doi.org/10.1016/j.ipm.2021.102722

[59] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

[60] Nasim Sonboli, Masoud Mansoury, Ziyue Guo, Shreyas Kadekodi, Weiwen Liu, Zijun Liu, Andrew Schwartz, and Robin Burke. 2021. librec-auto: A Tool for Recommender Systems Experimentation. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong (Eds.). ACM, 4584–4593. https://doi.org/10.1145/3459637.3482006

[61] Gábor Takács, István Pilászy, and Domonkos Tikk. 2011. Applications of the Conjugate Gradient Method for Implicit Feedback Collaborative Filtering. In *Proceedings of the Fifth ACM Conference on Recommender Systems* (Chicago, Illinois, USA) *(RecSys '11)*. Association for Computing Machinery, New York, NY, USA, 297–300. https://doi.org/10.1145/2043932.2043987

[62] Mucun Tian and Michael D. Ekstrand. 2020. Estimating Error and Bias in Offline Evaluation Results. In *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval (CHIIR '20)*. Association for Computing Machinery, New York, NY, USA, 392–396. https://doi.org/10.1145/3343413.3378004

[63] Chi Wang, Qingyun Wu, Markus Weimer, and Erkang Zhu. 2021. FLAML A Fast and Lightweight AutoML Library. In *MLSys*.

[64] Ting-Hsiang Wang, Xia Hu, Haifeng Jin, Qingquan Song, Xiaotian Han, and Zirui Liu. 2020. *AutoRec: An Automated Recommender System*. Association for Computing Machinery, New York, NY, USA, 582–584. https://doi.org/10.1145/3383313.3411529

[65] Lukas Wegmeth. 2022. The Impact of Feature Quantity on Recommendation Algorithm Performance: A Movielens-100K Case Study. https://doi.org/10.48550/arXiv.2207.08713 arXiv:2207.08713 [cs].

[66] Lukas Wegmeth and Joeran Beel. 2022. CaMeLS: Cooperative meta-learning service for recommender systems. In *Proceedings of the Perspectives on the Evaluation of Recommender Systems Workshop 2022*. CEUR-WS. https://ceur-ws.org/Vol-3228/paper2.pdf

[67] Sanjay Yadav and Sanyam Shukla. 2016. Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification. In *2016 IEEE 6th International Conference on Advanced Computing (IACC)*. 78–83. https://doi.org/10.1109/IACC.2016.25

[68] Lucas Zimmer, Marius Lindauer, and Frank Hutter. 2021. Auto-PyTorch Tabular Multi-Fidelity MetaLearning for Efficient and Robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), 3079 – 3090. also available under httpsarxiv.orgabs2006.13799.

Received 26 February 2023; revised 12 March 2009; accepted 5 June 2009