FairRecKit: A Web-based Analysis Software for Recommender Evaluations

Christine Bauer[*]
c.bauer@uu.nl
Utrecht University
Utrecht, The Netherlands

Lennard Chung
Utrecht University
Utrecht, The Netherlands

Aleksej Cornelissen
Utrecht University
Utrecht, The Netherlands

Isabelle van Driessel
Utrecht University
Utrecht, The Netherlands

Diede van der Hoorn
Utrecht University
Utrecht, The Netherlands

Yme de Jong
Utrecht University
Utrecht, The Netherlands

Lan Le
Utrecht University
Utrecht, The Netherlands

Sanaz Najiyan Tabriz
Utrecht University
Utrecht, The Netherlands

Roderick Spaans
Utrecht University
Utrecht, The Netherlands

Casper Thijsen
Utrecht University
Utrecht, The Netherlands

Robert Verbeeten
Utrecht University
Utrecht, The Netherlands

Vos Wesseling
Utrecht University
Utrecht, The Netherlands

Fern Wieland
Utrecht University
Utrecht, The Netherlands

ABSTRACT

FairRecKit is a web-based analysis software that supports researchers in performing, analyzing, and understanding recommendation computations. The idea behind *FairRecKit* is to facilitate the in-depth analysis of recommendation outcomes considering fairness aspects. With (nested) filters on user or item attributes, metrics can easily be compared across user and item subgroups. Further, (nested) filters can be used on the dataset level; this way, recommendation outcomes can be compared across several sub-datasets to analyze for differences considering fairness aspects. The software currently features five datasets, 11 metrics, and 21 recommendation algorithms to be used in computational experimentation. It is open source and developed in a modular manner to facilitate extension. The analysis software consists of two components: A software package (*FairRecKitLib*) for running recommendation algorithms on the available datasets and a web-based user interface (*FairRecKitApp*) to start experiments, retrieve results of previous experiments, and analyze details. The application also comes with extensive documentation and options for result customization, which makes for a flexible tool that supports in-depth analysis.

[*]Corresponding author

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Software and its engineering → Software libraries and repositories**; • **General and reference → Evaluation**.

KEYWORDS

resource, evaluation, analysis, recommender systems, toolkit, web-based, music, movies

ACM Reference Format:
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 *ACM SIGIR Conference on Human Information Interaction and Retrieval (CHIIR '23), March 19–23, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3576840.3578274

1 INTRODUCTION

Comparative experimentation is a central means in recommender systems research [19]. Yet, as a community, we are facing several challenges in this realm. First, despite progress in recent years, *reproducibility remains an issue* [2, 10]. Because already minor differences in parameters can yield incompatible results [3, 19], frameworks with standardized evaluation pipelines were introduced as a worthwhile path for the research community (e.g., [3, 6]). The various frameworks put different aspects into the loop. For instance, Anelli et al. [1] supplies a wide range of beyond-accuracy metrics. Bellogín and Said [3] provide guidelines to improve accountability. LensKit [6] does not only facilitate offline evaluation but has also been demonstrated useful for educational purposes. Second, there are *barriers to entry in the recommendation field for* *researchers* new to the field. LensKit moved from the Java implementation [9] to a Python toolkit [6] to provide "a more widely-used and easier-to-learn computational environment". Still, the required level of programming skills might be considered a barrier to entry for students outside the computer science field. While studying, for instance, the *impact* of recommender systems is interesting for students and researchers from a wide range of disciplines, the requirement to have substantial programming skills limits this kind of research to researchers from computing-oriented disciplines. As the recommender systems field is a multidisciplinary research field, it is important to reduce the barriers to entry from a wider set of disciplines.

Recently, there is an increasing interest to create fair recommender systems [5, 7]. Addressing the issue of fairness requires to *consider 'fairness' aspects in the evaluation.* However, it is often still unclear what is considered 'fair' [12] and what exactly needs to be evaluated [7]. One viable path to follow is to analyze metrics across various user or item subgroups [20] (as done, for example, in Ferraro et al. [11] and Ekstrand and Kluver [8]). Due to the existing research gaps, studying the fairness of recommender systems (still) requires a high level of exploratory analysis to delve into detail. For instance, studying aspects of fairness embraces exploring and comparing recommendation performance across various user or item subgroups, where various attributes can and need to be used for defining those subgroups. Furthermore, while we typically compare metrics across approaches, datasets, or subgroups, often little attention is paid to what items are recommended to users. Yet, looking into such details may provide insights into where biases exist and what kind of biases are at play. A tool facilitating such exploration would particularly help address fairness aspects where insights and inspiration can be drawn from such discoveries and observations.

This paper presents *FairRecKit*—a web-based *analysis software*—that aims to address these challenges. The idea behind *FairRecKit* is to facilitate experimentation considering fairness aspects. With (nested) filters on user or item attributes, metrics can easily be compared across user and item subgroups. *FairRecKit* currently features five datasets, 11 metrics, and 21 recommendation algorithms. The metrics and recommendation algorithms are integrated from existing libraries, and we use widely-used open datasets, which fosters the reproducibility (and comparability) aspect in recommender systems research. As a web-based *analysis software*, it aids researchers in performing, analyzing, and understanding recommendation computations. The web-based user interface allows researchers with a decent understanding of recommender systems—yet, not necessarily with extensive programming knowledge—to engage with comparative experimentation, reducing the barrier to entry. To the best of our knowledge, *FairRecKit* is the first toolkit to provide graphic access to recommender systems offline experimentation. Furthermore, *FairRecKit* allows for exploring the items that are recommended to individual users. Besides metadata on the items (e.g., genre, artist gender) integrated via the Last.fm API[1] and the MusicBrainz API[2], *FairRecKit* also integrates various features from the Spotify API[3] for music items (e.g., audio features such as danceability, audio snippets). *FairRecKit* is open source[4] and developed in a modular manner to facilitate extension. The software consists of two components: (i) A software package (*FairRecKitLib*) that is used to run recommendation algorithms on the available datasets and (ii) a web-based user interface (*FairRecKitApp*) where the researcher starts experiments and retrieves results of previous experiments. Figure 1 shows a global overview of the client architecture. *FairRecKit* is a flexible tool that supports researchers in analysis. It comes with extensive documentation; a video showing the system in action is available at the following URL: https://tinyurl.com/FairRecKitDemo.

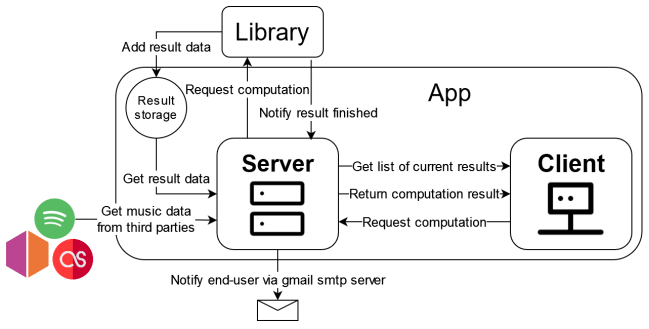


Figure 1: *FairRecKit*'s server-client architecture

In the following, we first describe the user interface and the functionality of *FairRecKitApp* (Section 2), followed by a description of *FairRecKitLib* (Section 3). We conclude this resource paper with a summary of the main functionalities, a discussion of limitations, and an outlook on opportunities for use and extension (Section 4).

2 COMPONENTS OF *FairRecKitApp*

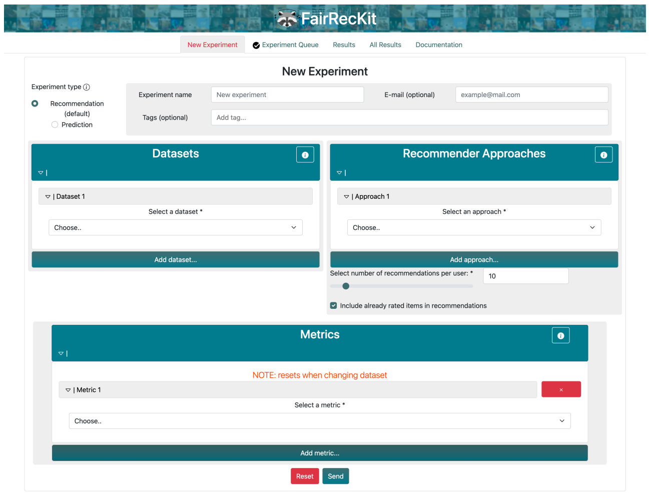
The web-based user interface *FairRecKitApp* consists of the following five parts, which are implemented as 'tabs': (i) New Experiment Tab, (ii) Experiment Queue Tab, (iii) Current Results Tab, (iv) All Results Tab, and the (v) Documentation Tab.

New Experiment Tab. This tab is used to configure new experiments (Figure 2). Thereby, one can choose from extensive settings to configure the experimentation. The configuration embraces the following four parts:

- *Datasets.* Currently, five datasets are stored and readily available to choose from (Table 1): LFM-1b [17], LFM-2b [18], and Last.fm 360K [4] for the music domain; MovieLens-100K and MovieLens-25M [14] for the movie domain. For each selected dataset, the train-test split (e.g., 80/20) and the type of split (random or temporal) can be specified. Further, multiple dataset filters are available (e.g., considering subgroups of users or items), whereby also nested filters can be set. For instance, to compare metrics across artist gender (as in Ferraro et al. [11]), it is advisable to only consider those data entries that contain artist gender information. In addition, a rating converter is optionally available to convert implicit

[1]https://www.last.fm/api
[2]https://musicbrainz.org/doc/MusicBrainz_API
[3]https://developer.spotify.com/documentation/web-api/

[4]The repository of *FairRecKitApp* can be accessed at https://github.com/FairRecKit/FairRecKitApp, the repository of *FairRecKitLib* at https://github.com/FairRecKit/FairRecKitLib.



Experiment type: [✓ Recommendation (default)] Prediction

Select number of recommendations per user: * 10

✓ Include already rated items in recommendations

Figure 2: Experiment Tab.

to explicit ratings; this allows for using recommendation approaches that are designed for explicit ratings while the dataset contains implicit ratings. Note that different (nested) filters or conversions may be configured for each selected dataset.

Table 1: Integrated datasets with domain and data origin.

Dataset name	Reference	Domain	Data origin
LFM-1b	[17]	music	Last.fm
LFM-2b	[18]	music	Last.fm
Last.fm 360K	[4][5]	music	Last.fm
MovieLens-100K	[14][6]	movies	MovieLens
MovieLens-25M	[14][7]	movies	MovieLens

- *Recommendation approaches.* In this setting, one or more of the 21 currently integrated recommendation approaches (Table 2) can be selected. Each approach comes with a set of default options that may be modified as required. Further, the number of item recommendations per user is specified, and whether to include items with known user ratings.
- *Metrics.* One or more metrics are available to choose from for the experiment (Table 3). Some metrics come with a set of default options that can be modified as required (e.g., setting the K in $P@K$). In addition, data filters can be applied to create subgroups of the metric results (e.g., $P@K_{female}$ for the $P@K$ achieved by considering items by female artists compared to the same metric for male artists ($P@K_{male}$) compared to the entire dataset ($P@K_{all}$), as done in Ferraro et al. [11]). This functionality is useful to compare metrics across item groups and user groups alike. Similar performance across subgroups is a cornerstone for fair recommender systems.

[5]https://www.upf.edu/web/mtg/lastfm360k
[6]https://grouplens.org/datasets/movielens/100k/
[7]https://grouplens.org/datasets/movielens/25m/

Table 2: Integrated recommendation approaches with the type of algorithmic approach and the source library.

Approach	Type	Library	Ref.
AlternatingLeastSquares	Matrix Factorization	Implicit	[13]
BiasedMF	Matrix Factorization	LensKit	[6]
ImplictMF	Matrix Factorization	LensKit	[6]
BayesianPersonalizedRanking	Matrix Factorization	Implicit	[13]
LogisticMatrixFactorization	Matrix Factorization	Implicit	[13]
BaselineOnlyALS	Matrix Factorization	Surprise	[15]
BaselineOnlySGD	Matrix Factorization	Surprise	[15]
SVD	Matrix Factorization	Surprise	[15]
SVDpp	Matrix Factorization	Surprise	[15]
NMF	Matrix Factorization	Surprise	[15]
ItemItem	Collaborative Filtering	LensKit	[6]
UserUser	Collaborative Filtering	LensKit	[6]
KNNBasic	Collaborative Filtering	Surprise	[15]
KNNBaselineALS	Collaborative Filtering	Surprise	[15]
KNNBaselineSGD	Collaborative Filtering	Surprise	[15]
KNNWithMeans	Collaborative Filtering	Surprise	[15]
KNNWithZScore	Collaborative Filtering	Surprise	[15]
CoClustering	Collaborative Filtering	Surprise	[15]
SlopeOne	Collaborative Filtering	Surprise	[15]
PopScore	Basic/Utility	LensKit	[6]
Random	Basic/Utility	LensKit	[6]

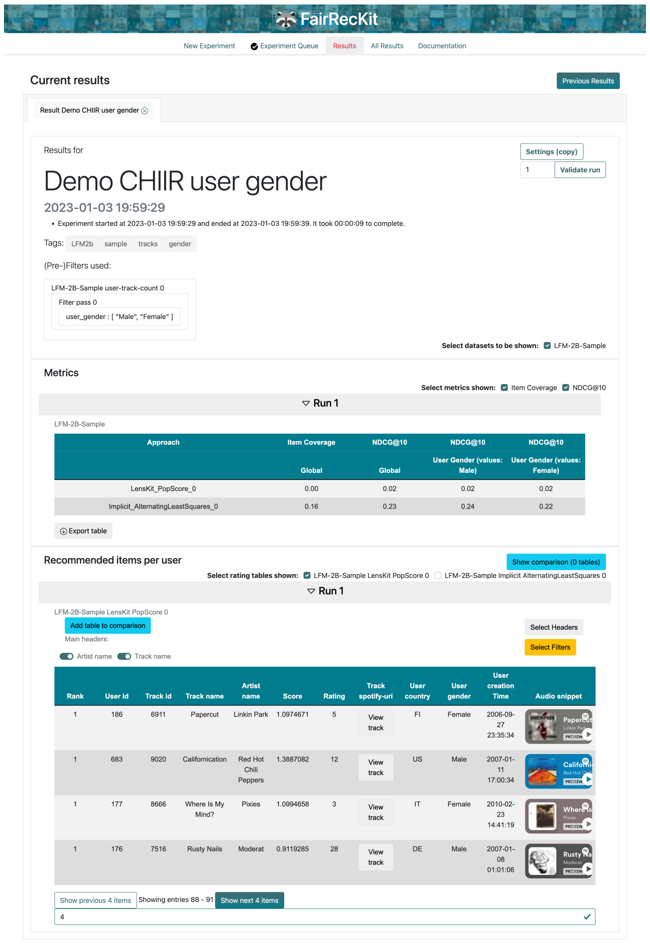
Table 3: Integrated metrics with metric type and source library.

Metric	Metric type	Library	Reference
HR@K	Classification	LensKit	[6]
P@K	Classification	LensKit	[6]
NDCG@K	Ranking	LensKit	[6]
MRR	Ranking	LensKit	[6]
RMSE	Rating	LensKit	[6]
MAE	Rating	LensKit	[6]
MAPE	Rating	rexmex	[16]
MSE	Rating	rexmex	[16]
ItemCoverage	Coverage	rexmex	[16]
UserCoverage	Coverage	rexmex	[16]

- *Metadata.* Supplying the experiment with a name and tags facilitates later retrieval from the list of the stored experimental results (Figure 4). Further, one may supply an e-mail address to be notified once the computation of an experiment has been completed.

Experiment Queue Tab. This tab shows the queue of all experiments that are waiting to be processed, with the currently processing one on top of the list. A progress bar indicates the progress of the current experiment. Entries in the queue may be canceled and removed from the queue.

Current Results Tab. This tab (Figure 3) provides all information about the currently loaded results, including the following:



Current results

Results for

Demo CHIIR user gender

2023-01-03 19:59:29

- Experiment started at 2023-01-03 19:59:29 and ended at 2023-01-03 19:59:39. It took 00:00:09 to complete.

Tags: LFM2b sample tracks gender

(Pre-)Filters used:

LFM-2B-Sample user-track-count 0
Filter pass 0
user_gender : ["Male", "Female"]

Select datasets to be shown: ✓ LFM-2B-Sample

Metrics

Select metrics shown: ✓ Item Coverage ✓ NDCG@10

Run 1

LFM-2B-Sample

Approach	Item Coverage	NDCG@10	NDCG@10	NDCG@10
	Global	Global	User Gender (values: Male)	User Gender (values: Female)
LensKit_PopScore_0	0.00	0.02	0.02	0.02
Implicit_AlternatingLeastSquares_0	0.16	0.23	0.24	0.22

Recommended items per user

Select rating tables shown: ✓ LFM-2B-Sample LensKit PopScore 0 ☐ LFM-2B-Sample Implicit AlternatingLeastSquares 0

Run 1

LFM-2B-Sample LensKit PopScore 0

Main headers: Artist name Track name

Rank	User id	Track id	Track name	Artist name	Score	Rating	Track spotify-uri	User country	User gender	User creation Time	Audio snippet
1	186	6911	Papercut	Linkin Park	1.0974671	5	View track	FI	Female	2006-09-27 23:35:34	Papercut Linkin Park PREVIEW
1	683	9020	Californication	Red Hot Chili Peppers	1.3887082	12	View track	US	Male	2007-01-11 17:00:34	Californi Red Hot Chili PREVIEW
1	177	8666	Where Is My Mind?	Pixies	1.0994658	3	View track	IT	Female	2010-02-23 14:41:19	Where Is Pixies PREVIEW
1	176	7516	Rusty Nails	Moderat	0.9119285	28	View track	DE	Male	2007-01-08 01:01:06	Rusty Na Moderat PREVIEW

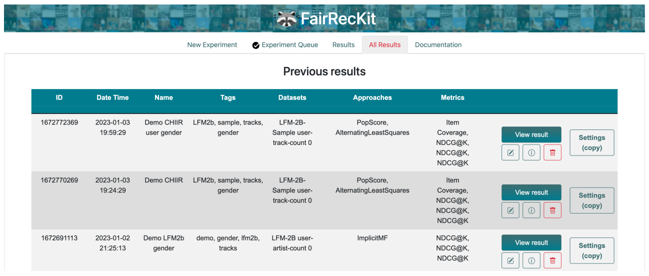
Showing entries 88 - 91

4

Figure 3: Current Results Tab.

- *Experiment results.* The specified metrics are displayed in tables, separately for each dataset-approach pair. If an experiment has been validated (thus, it had multiple runs), each run will be displayed individually. Finally, each table may be exported as a *TSV* file.
- *User recommendations.* This table shows a (long) list of user-item combinations with a calculated recommendation (or prediction) rating and the rank. Further, this table also the user's (original) rating if available for an entry. Via a menu, additional columns can be added; for instance, more information about a user or item may be displayed (e.g., user age or artist gender). Further, filters may be applied to the table (e.g., showing only items by female artists). Another functionality is the Spotify integration for every music dataset matrix that supports it. If selected, the table may display the album that an item (track) belongs to and a short audio snippet.[8] One may modify the number of entries that are loaded per page of the table,[9] and the table pages may be navigated via the 'Show next' and 'Show previous' buttons. Initially, the table is sorted by the first column (rank). Sorting by other properties may be activated by clicking on the respective column header (ascending or descending).
- *Validation.* Experimental results may be validated by performing the computational experiment with the same settings for an additional number of runs. Validation results can be accessed in the same tab as the original results.
- *Table selection.* Experiments including various datasets, approaches, and runs, will deliver enormous result tables. To counteract the information clutter and provide focus, check boxes are included to hide/show certain datasets, approaches, or runs.
- *Table comparison.* This functionality allows a comparison of the recommendation results for selected users. To this end, the researcher may click the "Add table to comparison" button when viewing the *User Recommendations* of a specific user; this will add the respective table to the comparison view. This can be repeated for several tables across several users respectively, which will add the respective tables to the comparison view. Clicking the "show comparison" button will open a window containing the tables that have been added to the comparison view. Tables can be re-positioned with 'drag &drop'; or removed from the comparison view.

All Results Tab. This tab (Figure 4) shows a list of all previously computed experiments stored on the server, along with the respective metadata, and the specified datasets, approaches, and metrics. One may edit the metadata, view details about an experiment's properties, load it into the *Current Result Tab*, or delete it. Further, there is an option to copy an experiment's configuration into the *New Experiment Tab* and run a new experiment, whereby the configuration may be adapted before initiating the new experiment.



Previous results

ID	Date Time	Name	Tags	Datasets	Approaches	Metrics	
1672772369	2023-01-03 19:59:29	Demo CHIIR user gender	LFM2b, sample, tracks, gender	LFM-2B-Sample user-track-count 0	PopScore, AlternatingLeastSquares	Item Coverage, NDCG@K, NDCG@K, NDCG@K	View result Settings (copy)
1672770269	2023-01-03 19:24:29	Demo CHIIR	LFM2b, sample, tracks, gender	LFM-2B-Sample user-track-count 0	PopScore, AlternatingLeastSquares	Item Coverage, NDCG@K, NDCG@K, NDCG@K	View result Settings (copy)
1672691113	2023-01-02 21:25:13	Demo LFM2b gender	demo, gender, lfm2b, tracks	LFM-2B user-artist-count 0	ImplicitMF	NDCG@K, NDCG@K, NDCG@K	View result Settings (copy)

Figure 4: All Results Tab.

Documentation Tab. This tab provides an overview of the terminology used in the application. Whenever a researcher does not recognize a certain component, whether it is a dataset, metric, filter, approach, or functionality, the documentation provides the information to utilize the respective component to its full extent. The *Documentation Tab* also features a navigation menu to quickly find a component of interest.

3 COMPONENTS OF THE *FairRecKitLib*

The library part of the software is an extensive Python library, which is used to run approaches on datasets and to evaluate the

[8]Note that the Spotify features are hidden by default (because loading all these snippets is slow) but can be enabled by clicking on the 'Show snippets' check box.
[9]By default, the table's first 10 entries are loaded.

results. In this section, we describe its most essential parts; for more details, we kindly refer to the documentation[10]. The main components of *FairRecKitLib* are the following: (i) Core functionality, (ii) Datasets and matrices, (iii) Data pipeline, (iv) Model pipeline, (v) Evaluation pipeline, (vi) Experiment pipeline, and the (vii) Recommender system.

We carefully considered the advantages and disadvantages of existing libraries to find a good mix. The configuration file (layout) for an experiment was inspired by the one used in ELLIOT [1]. The recommendation algorithm interface was inspired by LensKit [6, 9], which also allowed us to easily incorporate Implicit [13] and Surprise [15].

Core functionality. The core of the library contains the functionality that is needed throughout the pipelines: parsing input, managing threads, configurations, events, input/output (I/O), and the base for pipelines. Parsing the input is a vital part of the library as all functions require their input to be in a correct format and have correct values to run experiments. Therefore, the library includes many parsers to check input before it is passed to their respective functions. Managing threads is useful when working with heavy computations on big data where these computations would otherwise block the recommender system from being used while active. The base class for the pipelines is located in the core of the library. There are three different pipelines, which are all based on the base pipeline in the core: the data, model, and evaluation pipelines. An additional pipeline connects the previously mentioned pipelines using a data transition which is the experiment pipeline.

Datasets and matrices. Various datasets are first processed (once) before they become available in a data registry. While processing, a configuration file is generated, describing the available (event) tables and matrices, which is similar to a database definition. Moreover, additional user-item matrices can be generated when a dataset includes an event table, as that table contains user events that have happened (e.g., a user consumed a specific item at a certain point in time).

The (pre-)processing step makes sure that each dataset is appropriately processed and converted into an abstract standard format. This makes them easier to use not only in the library itself but also in the *FairRecKitApp*. Further, it facilitates adding new datasets without major challenges.

Data pipeline. The data pipeline is the first step in running an experiment. Here, the data is processed before running approaches on it. The data configuration defines one or more dataset matrices that need to be used in the experiment, which are loaded in this pipeline. One or more filter passes are applied when provided to create a subset, where each filter pass applies one or more filters sequentially. The ratings in the dataset matrix can be converted to be within a specific range, and finally, the resulting matrix will be split into a train set and a test set. Then, it is ready for the model pipeline.

Model pipeline. The model pipeline is the second step in running an experiment. Here, the model configuration defines which approaches are run on the train set. The *FairRecKitLib* uses three other libraries to provide the algorithmic approaches: Implicit [13], LensKit [6, 9], and Surprise [15]. The test set, which was created together with the train set, will be used to determine for which users to produce a number of item recommendations or for which user-item pairs to compute rating predictions. The computed ratings are stored in a file incrementally; and when finished, it will be ready for evaluation in the next pipeline.

[10]https://fairreckit.github.io/FairRecKitLib/

Evaluation pipeline. The evaluation pipeline is the third, yet optional, step in running an experiment and is executed when the model pipelines finish. According to the evaluation configuration, a variety of metrics are used to compute and measure the performance of the used dataset-approach pairs. Two libraries are used to provide these metrics, namely LensKit [6] and rexmex [16]. Most of these metrics will use the computed ratings together with the test set to validate the outcomes, whereas some (e.g., user or item coverage) require the train set instead of the test set. In addition, these sets can be filtered using multiple passes, and hence, they can be used to compute metrics for a specific subgroup instead of the global performance.

Experiment pipeline. The experiment pipeline connects the three previously mentioned pipelines to compute an experiment for one or more runs. Therefore, the experiment configuration is a combination of the data, model, and evaluation configurations. First, the data pipeline is run for each selected dataset matrix. Next, each generated transition result of the data pipeline is fed into the model pipelines for all the selected approaches. Lastly, each time the model pipelines finish a data transition result, these are forwarded right away to run through the evaluation pipelines to compute all selected metrics. This process is then repeated for the number of runs that are specified alongside the configuration of the experiment. This happens on a separate thread so that the experiment run(s) can be canceled at any time.

Recommender system. The top-level API of the library is the recommender system. The system is initialized by providing it with a directory where the datasets are stored and a directory where to store the experiment results. The recommender system can be queried for the availability of datasets, approaches, and metrics. Availability of data modification operations can also be requested; namely data filters, matrix rating converters, and matrix train-test splitters. Lastly, the recommender system provides functionality to start a new experiment, validate an existing experiment for an additional number of runs, and cancel an active computation.

4 CONCLUSION

To sum up, *FairRecKit* is a web-based analysis software to support researchers in performing, analyzing, and understanding recommendation computations. The software currently features five datasets, 11 metrics, and 21 recommendation algorithms to be used in experimentation. It is open source and developed in a modular manner to facilitate extension, which is strongly encouraged. Opportunities lie in providing a wider scope of metrics. Thereby, embracing beyond-accuracy metrics would be a strong asset. To provide the opportunity to analyze for a wider range of fairness aspects, an enrichment of the datasets with additional metadata would be beneficial. Further, expanding the scope of domains is a viable option. Currently, the focus lies on the music domain as, for instance, Spotify audio features are integrated. Beyond serving research purposes, *FairRecKit* is a software resource that might also be relevant for teaching.

ACKNOWLEDGMENTS

FairRecKit has been developed by students within the Software Project course of the Bachelor program Computer Science at Utrecht University, commissioned by Christine Bauer. The project was funded by the Department of Information and Computing Sciences of Utrecht University.

REFERENCES

[1] Vito Walter Anelli, Alejandro Bellogín, 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* (Virtual Event, Canada) *(SIGIR '21)*. ACM, New York, NY, USA, 2405–2414. https://doi.org/10.1145/3404835.3463245

[2] Joeran Beel, Corinna Breitinger, Stefan Langer, Andreas Lommatzsch, and Bela Gipp. 2016. Towards reproducibility in recommender-systems research. *User Modeling and User-Adapted Interaction* 26, 1 (2016), 69–101. https://doi.org/10.1007/s11257-016-9174-x

[3] Alejandro Bellogín and Alan Said. 2021. Improving accountability in recommender systems research through reproducibility. *User Modeling and User-Adapted Interaction* 31, 5 (2021), 941–977. https://doi.org/10.1007/s11257-021-09302-x

[4] Òscar Celma. 2010. *Music Recommendation.* Springer Berlin Heidelberg, Berlin, Heidelberg, Germany, Chapter 3, 43–85. https://doi.org/10.1007/978-3-642-13287-2_3

[5] Karlijn Dinnissen and Christine Bauer. 2022. Fairness in music recommender systems: a stakeholder-centered mini review. *Frontiers in Big Data* 5, Article 913608 (2022), 9 pages. https://doi.org/10.3389/fdata.2022.913608

[6] 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* (Virtual Event, Ireland) *(CIKM '20)*. ACM, New York, NY, USA, 2999–3006. https://doi.org/10.1145/3340531.3412778

[7] Michael D Ekstrand, Anubrata Das, Robin Burke, and Fernando Diaz. 2022. Fairness in Information Access Systems. *Foundations and Trends® in Information Retrieval* 16, 1–2 (2022), 1–177. https://doi.org/10.1561/1500000079

[8] Michael D. Ekstrand and Daniel Kluver. 2021. Exploring author gender in book rating and recommendation. *User Modeling and User-Adapted Interaction* 31, 3 (2021), 377–420. https://doi.org/10.1007/s11257-020-09284-2

[9] Michael D. Ekstrand, Michael Ludwig, Joseph A. Konstan, and John T. Riedl. 2011. Rethinking the Recommender Research Ecosystem: reproducibility, openness, and LensKit. In *Proceedings of the 5th ACM Conference on Recommender Systems* (Chicago, IL, USA) *(RecSys '11)*. ACM, New York, NY, USA, 133–140. https://doi.org/10.1145/2043932.2043958

[10] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. 2021. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research. *ACM Transactions on Information Systems* 39, 2, Article 20 (jan 2021), 49 pages. https://doi.org/10.1145/3434185

[11] Andrés Ferraro, Xavier Serra, and Christine Bauer. 2021. Break the Loop: Gender Imbalance in Music Recommenders. In *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval* (Virtual Event) *(CHIIR '21)*. ACM, New York, NY, USA, 249–254. https://doi.org/10.1145/3406522.3446033

[12] Andrés Ferraro, Xavier Serra, and Christine Bauer. 2021. What is fair? exploring the artists' perspective on the fairness of music streaming platforms. In *Human-Computer Interaction – INTERACT 2021.* Lecture Notes in Computer Science, Vol. 12933. Springer International Publishing, Cham, Germany, 562–584. https://doi.org/10.1007/978-3-030-85616-8_33

[13] Ben Frederickson. 2017. Implicit package. https://implicit.readthedocs.io/en/latest/index.html.

[14] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4, Article 19 (Dec. 2015), 19 pages. https://doi.org/10.1145/2827872

[15] Nicolas Hug. 2015. Surprise package. https://surprise.readthedocs.io/en/stable/index.html.

[16] Benedek Rozemberczki, Sebastian Nilsson, and Piotr Grabowski. 2022. rexmex package. https://rexmex.readthedocs.io/en/latest/index.html.

[17] Markus Schedl. 2016. The LFM-1b Dataset for Music Retrieval and Recommendation. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval* (New York, NY, USA) *(ICMR '16)*. ACM, New York, NY, USA, 103–110. https://doi.org/10.1145/2911996.2912004

[18] Markus Schedl, Stefan Brandl, Oleg Lesota, Emilia Parada-Cabaleiro, David Penz, and Navid Rekabsaz. 2022. LFM-2b: A Dataset of Enriched Music Listening Events for Recommender Systems Research and Fairness Analysis. In *ACM SIGIR Conference on Human Information Interaction and Retrieval* (Regensburg, Germany) *(CHIIR '22)*. ACM, New York, NY, USA, 337–341. https://doi.org/10.1145/3498366.3505791

[19] Nasim Sonboli, Robin Burke, Zijun Liu, and Masoud Mansoury. 2020. Fairness-Aware Recommendation with Librec-Auto. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20)*. ACM, New York, NY, USA, 594–596. https://doi.org/10.1145/3383313.3411525

[20] Eva Zangerle and Christine Bauer. 2022. Evaluating recommender systems: survey and framework. *Comput. Surveys* 55, 8, Article 170 (2022), 38 pages. https://doi.org/10.1145/3556536