MANILA: A Low-Code Application to Benchmark Machine Learning Models and Fairness-Enhancing Methods

Giordano d'Aloisio
giordano.daloisio@univaq.it
University of L'Aquila
L'Aquila, Italy

Abstract

This paper presents MANILA, a web-based low-code application to benchmark machine learning models and fairness-enhancing methods and select the one achieving the best fairness and effectiveness trade-off. It is grounded on an Extended Feature Model that models a general fairness benchmarking workflow as a Software Product Line. The constraints defined among the features guide users in creating experiments that do not lead to execution errors. We describe the architecture and implementation of MANILA and evaluate it in terms of expressiveness and correctness.

CCS Concepts

• **Software and its engineering** → **Extra-functional properties**; *Designing software*; • **Computing methodologies** → *Machine learning*.

Keywords

Fairness, Machine Learning, Low Code, Benchmarking

ACM Reference Format:
Giordano d'Aloisio. 2025. MANILA: A Low-Code Application to Benchmark Machine Learning Models and Fairness-Enhancing Methods. In *Companion Proceedings of the 33rd ACM Symposium on the Foundations of Software Engineering (FSE '25), June 23–27, 2025, Trondheim, Norway*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Machine learning (ML) systems are extensively employed in many domains nowadays. If we consider the impact that those applications have in our lives, it is clear how ensuring that those systems are of *high* quality is of paramount importance. However, the quality of ML systems is not limited to their effectiveness (i.e., prediction correctness) but also to other new quality attributes [13, 29, 31]. Among those, *fairness* (i.e., the absence of prejudice or discrimination of a system towards individuals or groups identified by a set of sensitive features [30]) emerged as one of the most critical quality attributes of ML systems as also highlighted by some of the 17 Sustainable Development Goals (SDG) of the United Nations [33] like SDG 5 (Gender Equality) or SDG 10 (Reduced Inequalities).

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

In recent years, many low-code methods have been developed to automate some phases of ML system development and help less technical users, e.g., [22, 24, 37]. However, a low-code approach that also considers the model's fairness and guides less technical users in the benchmark of different ML models and fairness-enhancing methods is still missing. Indeed, there is a need for low-code approaches to help democratize the quality-based development of ML systems [26, 32]

With this paper, we aim to fill this gap by proposing MANILA, a web-based low-code application to benchmark ML models and fairness-enhancing methods and select the combination that achieves the best fairness-effectiveness trade-off. The theoretical foundation of MANILA is an Extended Feature Model (ExtFM) that models a general fairness benchmarking workflow as a Software Product Line (SPL). The constraints among the features in the ExtFM have been reported in the web application and guide users in the definition of experiments that are always executable - i.e., they do not lead to execution errors. For space constraints, we leave details on the theoretical foundations of MANILA and to the ExtFM to our previous work [17].

MANILA can be employed by practitioners and researchers who have knowledge of ML but lack expertise in the fairness domain and want to benchmark different combinations of ML models and fairness-enhancing methods to identify the best one.

MANILA is publicly available in the SoBigData research infrastructure [23][1], and its source code is available on GitHub [15]. A documentation site presenting a hands-on tutorial on MANILA is also publicly available [14]. A video demonstration of MANILA is available online [16].

2 MANILA

MANILA is a low-code application to benchmark ML models and fairness-enhancing methods and to select the one that offers the best effectiveness and fairness trade-off. As outlined in [17], each fairness benchmarking workflow comprises a set of features acting as *variation points*, differentiating them from one another. For this reason, we can think of this family of experiments as a Software Product Line (SPL) specified by an Extended Feature Model [17].

Figure 1 details a high-level picture of MANILA, where each rounded box represents a step in the fairness benchmarking workflow, while square boxes represent artifacts. MANILA has been implemented as a low-code web-based application through which all the steps of the fairness benchmarking workflow can be performed. Near each artifact, we report the technologies involved in its implementation.

[1] https://sobigdata.d4science.org/group/sobigdata.it/manila-univaq (registration needed)

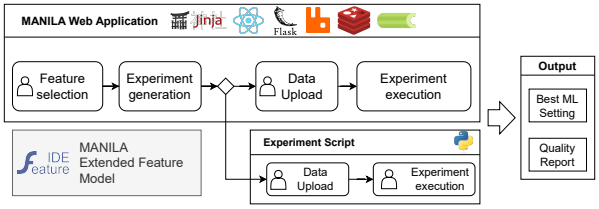

Figure 1: MANILA high-level overview

The first step in the development process is feature selection, in which the user selects all the components of the fairness benchmarking workflow through a dedicated web form. Next, a Python script implementing the benchmarking experiment is automatically generated from the selected features. The constraints defined among features guide the user in defining experiments that are always executable (i.e., do not lead to execution errors) [17]. The experiment performs a grid search [27] over the selected combinations of ML models and fairness-enhancing methods and computes the selected fairness and effectiveness metrics. The experiment can be downloaded for local execution or directly executed on the server. After its execution, it returns: *i)* a quality report reporting for each fairness-enhancing method and ML model the related metrics; *ii)* the best combination identified using the selected trade-off strategy.

As described in [17], the foundation of MANILA is an Extended Feature Model (ExtFM) that models the fairness benchmarking workflow as an SPL. The ExtFM is a template of all possible experiments a user can perform and guides them through selecting features comprising a fairness benchmarking workflow. We used the ExtFM as a formalism to reason about the different relationships and constraints among features before implementing them in the web application.

The web application has been implemented using the React Javascript library [6] for the frontend and the Flask Python library [4] for the backend. Moreover, we employ the Celery worker engine [2], with the RabbitMQ message broker [5] and the Redis database [7], to run the experiments asynchronously on the server. This way, we avoid overloading the server in case of multiple experiment runs.

2.1 Feature Selection

The feature selection step has been implemented in MANILA through a web form that includes all the features and the constraints defined in the ExtFM. The form consists of seven subsections, each corresponding to one of the macro features defined in the ExtFM. Each subsection includes all the concrete (i.e., non-abstract [9]) child features related to the respective macro feature in the ExtFM. The sections in the form are the following. ① **Dataset:** This section contains features to specify information about the dataset (e.g., the label or sensitive features). ② **Data Scaler:** This section includes all data scaler algorithms from the *scikit-learn* library [35] to pre-process data before training a model. ③ **ML Model:** This section allows the users to select the ML models to benchmark. All the ML models for classification and regression from the *scikit-learn* library are listed.

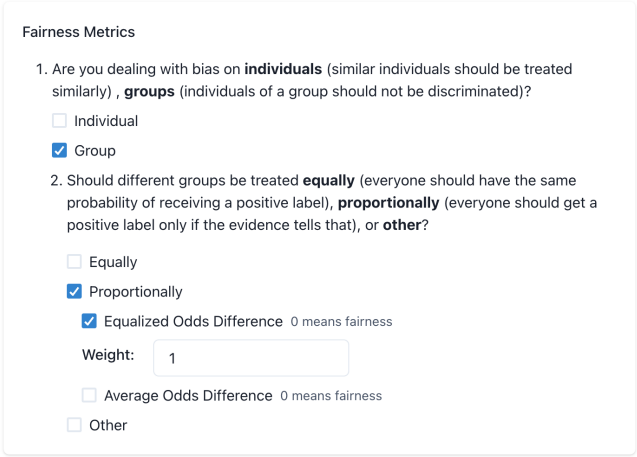

Figure 2: Fairness metric selection

④ **Fairness Methods:** This section allows the selection of the fairness-enhancing methods to benchmark. We included all models from `aif360` [10] and `Fairlearn` [12] libraries and the *Debiaser for Multiple Variables* algorithm [19]. ⑤ **Metrics:** This section includes all metrics that can be employed to evaluate the effectiveness and fairness of a given ML model and fairness-enhancing method combination. Concerning effectiveness metrics, we included all the classification and regression metrics available in the `scikit-learn` library. For fairness assessment, we included all the metrics from the `aif360` library. In addition, given the magnitude of fairness definitions and metrics available [30], following [40], we included a set of questions to help the data scientist select the fairness metrics more suited for their use case. The questions are reported in Figure 2. ⑥ **Tradeoff Strategy:** This section lists the strategies that can be adopted to identify the best ML model and fairness-enhancing method combination. In particular, the best setting can be identified by MANILA using an aggregation function of the selected metrics (e.g., mean, weighted sum, or harmonic mean) or through a Pareto-front of the best solutions [21]. ⑦ **Validation:** This last section allows the selection of strategies for cross-validation of the different ML models and fairness-enhancing methods combination.

Children with an *alternative* relationship in the ExtFM [9] are implemented in the form either through a radio group or by a logical condition among fields. In all other cases, features in the ExtFM have been implemented as checkbox fields in the form. Additional attributes related to the features (like the *Label Name* or *Positive Value* fields) have been implemented as text fields, which may be mandatory or not, depending on the case. Additionally, the cross-tree constraints defined in the ExtFM have been implemented as logical constraints among the different form fields. Figure 3 shows an example of such constraints. In the figure, it can be seen how the *Regression* field has been disabled. This is due to two reasons: first, the *Classification* ML task has already been selected, and second, the *Regression* task is incompatible with the fairness methods included so far. Hence, since fairness-enhancing methods have already been selected in another section of the form, ML methods for regression can not be selected. Otherwise, they would lead to a non-executable

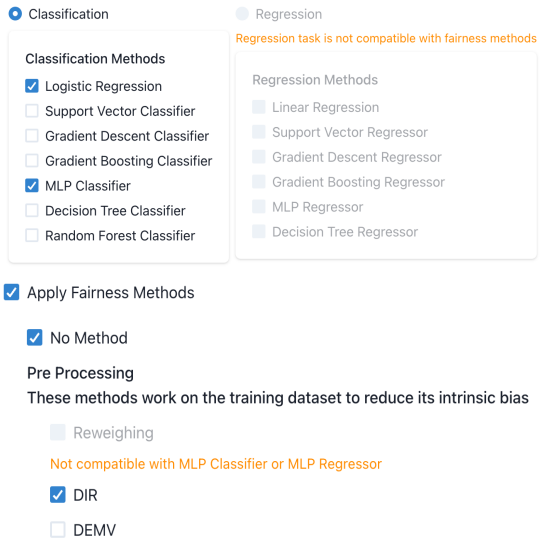

Figure 3: Example of web form cross-tree constraints

experiment. This constraint is also shown to the user through a message reporting that "*Regression task is incompatible with fairness methods.*" In addition, note how the *Reweighing* fairness method has been disabled as well. This is because the *Reweighing* method is not compatible with the *MLP Classifier* ML method that has already been selected. This constraint is also reported to the user, reporting that Reweighing is "*not compatible with MLP Classifier or MLP Regressor.*"

Finally, the form enables users to upload their dataset for running the generated experiment on the server. Below this field, two buttons are available: one for downloading the generated code and the other for executing the experiment on the server.

2.2 Experiment Generation and Execution

After selecting a set of features that meet all the form's constraints, the experiment can be executed. As shown in Figure 1, the experiment can be either downloaded for local execution or run directly on the server. The experiment performs a grid search across all combinations of the selected ML models and fairness-enhancing methods, computing the selected metrics for each combination. It is worth noticing how the space of the search (i.e., the possible combinations of ML models and fairness-enhancing methods) is always relatively small, given the constraints imposed by the ExtFM.

If the user chooses to download the experiment, MANILA generates the corresponding Python implementation relying on the Jinja template engine [34]. Additionally, it generates the `environment.yml` file needed to create the *conda* environment with all the required libraries [1]. The experiment can be executed locally by running the following command:

```
$ python main.py -d <DATASET PATH>
```

Otherwise, it can be called through a REST API or any other interface such as a desktop application or a Scientific Workflow Management System like *KNIME* [11, 28]. This generality of our experimental workflow makes it very flexible and suitable for many use

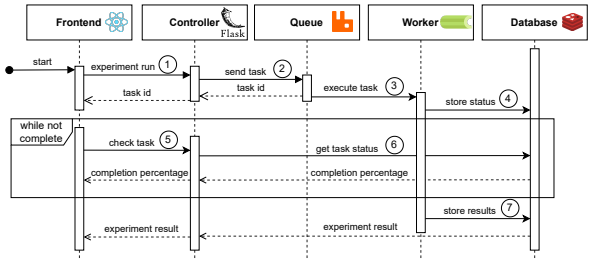

Figure 4: Online experiment execution

cases. After the execution, the code returns the quality report in CSV format. In addition, the best ML model (identified using the selected trade-off strategy) is trained with the full input dataset and by applying the best fairness-enhancing method. The ML model returned by the experiment is saved as a `dill` file [3]. We have chosen this format since it is an extension of the standard `pickle` format that enables the storage of more complex objects.

On the contrary, if the user chooses to run the experiment online, it is executed asynchronously using the Celery task queue system [2]. Figure 4 shows the online execution process. First, the frontend makes a call to the app controller through its REST API (step ① in Figure 4). The controller sends the task to a RabbitMQ queue and returns the task ID to the frontend (step ② in Figure 4). Next, the task is executed by a Celery worker, and its status is stored in a Redis database (steps ③ and ④ in Figure 4). At the same time, until the experiment execution is not completed, the frontend periodically asks the controller about the task status (steps ⑤ and ⑥ in Figure 4). If the experiment is still in progress, a completion percentage is returned. Otherwise, when the execution is completed, the Celery worker saves the results on Redis (step ⑦ in Figure 4), and the results are returned to the frontend. The results are shown in a dedicated page. The result page shows the metrics in a bar chart and in a tabular way. Additionally, it allows the download of the fully trained best ML model in `dill` format and the computed metrics as a CSV file, respectively.

3 Evaluation

In this section, we describe the initial evaluation we performed on MANILA. Following previous work [40], we evaluate MANILA in terms of *expressiveness* and *correctness*. In particular, we reproduce with MANILA the experimental evaluation described in [19] to evaluate the DEMV bias mitigation algorithm. More in detail, the *expressiveness* is assessed by proving that MANILA can replicate the selected fairness evaluation. In contrast, *correctness* is assessed by showing that the results obtained with the experiments generated by MANILA are comparable to the original ones.

We replicate three specific experiments performed on DEMV (i.e., the ones shown in Tables 16, 18, and 19 of [19]). We have chosen to replicate these experiments among all the ones conducted in the paper because they provide the highest combination of ML methods, fairness-enhancing methods, and metrics.

The *expressiveness* is evaluated by assessing if MANILA can properly reproduce the described experiments. The *correctness* is

Table 1: *p-values* of the Kruskal-Wallis H test for each experiment

	SP	AO	ZO Loss	DI	Acc	H-Mean
Exp. 1	0.84	0.08	0.84	0.45	0.38	0.53
Exp. 2	0.10	0.71	0.31	0.43	0.27	0.71
Exp. 3	1.00	0.57	0.16	0.96	0.72	0.75

evaluated by assessing if the results of the experiments generated by MANILA are close to the ones reported in the original experiments. More in detail, following previous work [40], the results of the generated experiments should be within the standard deviation range of the results reported in Tables 16, 18, and 19, and there should not be a statistically significant difference between the results. The replication package of the experiments is available in our Github repository [15].

Additionally, we plan to evaluate MANILA's *usefulness* (i.e., how much MANILA is perceived as useful) and *usability* (i.e., how much MANILA is easy to use) by performing a user evaluation involving people with diverse level of expertise in Software Engineering and Machine Learning-i.e., computer science master students, researchers, and practitioners. Following standard guidelines [36], we plan to ask each group of evaluators to perform two fairness benchmarking experiments: one using Python code and one using MANILA. Eventually, we will collect feedback on the positive and negative aspects of conducting the benchmarking process in Python and with MANILA. Moreover, we plan to compare the capabilities of MANILA against similar baseline approaches [25, 40].

Expressiveness Evaluation. We were able to correctly reproduce all the experiments. In particular, following the steps described in Section 2, we first specified the features of the experiments (i.e., ML models, fairness-enhancing methods, and metrics) from the graphical interface of MANILA. Next, we downloaded the generated codes to execute them locally. Finally, we ran the experiments to obtain the results. In total, we generated and executed ten different experiments (i.e., one for each input dataset). Being a low-code platform, MANILA does not require to write any line of code to implement the given experiments. In contrast, the original experiments required almost 200 lines of code, as seen in the repository linked in the original paper [19].

Correctness Evaluation. Table 1 reports, for each metric of each experiment, the *p-values* of the non-parametric Kruskal-Wallis H test performed between the results of the original experiments and the ones obtained by executing the code generated by MANILA. From the table, it can be seen that all the *p-values* are > 0.05, meaning that all the metrics obtained by running the code generated by MANILA are not statistically different from the original ones. In addition, Figure 5 reports a comparison of the aggregated h-means [20] of the three experiments. As shown, on average, the results of the three experiments generated by MANILA are very close to the original ones and are within the standard deviation range.

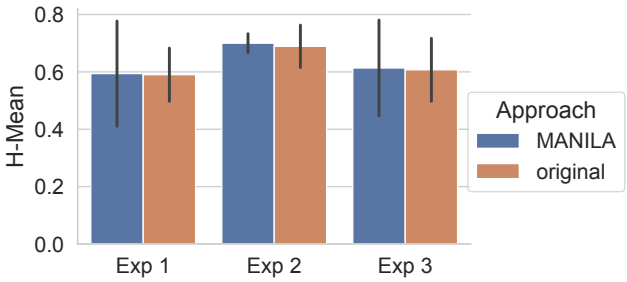

Figure 5: Aggregated H-Means of the original experiments and MANILA's ones

4 Related Work

Low-code approaches like *Aequitas* [38] Google's *What-If Tool* [39], Themis [8], and MODNESS [18] have been proposed to ease and democratize the fairness assessment of ML models. However, they do not allow to benchmark the combination of different ML models and fairness-enhancing methods.

A similar search-based approach to identify the best combination of the ML classifier and the fairness enhancing method is the *Fairkit-learn* Python library proposed by Johnson and Brun [25]. The library employs a grid-search approach to identify the optimal combination of fairness-enhancing methods and ML classifiers. In contrast, our work proposes a low-code application aimed at democratizing this process, making it more accessible for less technical users [26]. Additionally, MANILA provides guidance for defining and generating executable experiments, ensuring they don't lead to execution errors.

Yohannis and Kolovos propose a model-driven framework for bias assessment and mitigation [40]. The authors have first defined a meta-model to represent bias measurement and mitigation experiments. Then, they defined a Domain Specific Language (DSL) to specify such evaluations. The proposed tool automatically generates a Python script implementation from a script model defined using such a DSL. However, their approach does not guide the user in the definition of experiments that are always executable. By relying on the ExtFM formalism, MANILA fills this gap. Additionally, MANILA allows the online execution of the experiment. Thus, it does not require the user to execute the evaluation manually.

5 Conclusion

In this paper, we presented MANILA, a low-code web application that can be employed by practitioners and researchers with less knowledge of fairness to benchmark different ML models and fairness-enhancing method combinations. MANILA is grounded on the ExtFM formalism, which models a general fairness benchmarking workflow as an SPL where the variation points are the different components of the experiment, such as the ML models, fairness-enhancing methods, metrics, or trade-off strategies. The constraints defined among features guide users in creating always executable experiments. We described the system's architecture and evaluated it in terms of *expressiveness* and *correctness* by replicating a real experimental evaluation.

Acknowledgments

This work is partially supported by "FairLM: Addressing and Improving the Intersectional Bias of Large Language Models" a project founded by the University of L'Aquila, 2025.

References

[1] [n. d.]. Conda website. https://docs.conda.io/
[2] 2025. Celery Library. https://docs.celeryq.dev/en/stable
[3] 2025. Dill documentation. https://dill.readthedocs.io/
[4] 2025. Flask Library. https://flask.palletsprojects.com/
[5] 2025. RabbitMQ Website. https://www.rabbitmq.com/
[6] 2025. React Library. https://react.dev/
[7] 2025. Redis Website. https://redis.io/
[8] Rico Angell, Brittany Johnson, Yuriy Brun, et al. 2018. Themis: automatically testing software for discrimination. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (Esec/fse 2018)*. Association for Computing Machinery, New York, NY, USA, 871–875. https://doi.org/10.1145/3236024.3264590
[9] Sven Apel, Don Batory, Christian Kästner, et al. 2016. *Feature-oriented software product lines*. Springer.
[10] R. K. E. Bellamy, K. Dey, M. Hind, et al. 2019. AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development* 63, 4/5 (2019), 4:1–4:15. https://doi.org/10.1147/jrd.2019.2942287 Conference Name: IBM Journal of Research and Development.
[11] Michael R. Berthold, Nicolas Cebron, Fabian Dill, et al. 2009. KNIME - the Konstanz Information Miner: Version 2.0 and Beyond. *SIGKDD Explor. Newsl.* 11, 1 (2009), 26–31. https://doi.org/10.1145/1656274.1656280
[12] Sarah Bird, Miro Dudík, Richard Edgar, et al. 2020. *Fairlearn: A toolkit for assessing and improving fairness in AI*. Technical Report Msr-tr-2020-32. Microsoft. https://www.microsoft.com/en-us/research/publication/fairlearn-a-toolkit-for-assessing-and-improving-fairness-in-ai/
[13] Jan Bosch, Helena Holmström Olsson, and Ivica Crnkovic. 2021. Engineering AI Systems: A Research Agenda. https://doi.org/10.4018/978-1-7998-5101-1.ch001 ISBN: 9781799851011 Pages: 1-19 Publisher: IGI Global.
[14] Giordano d'Aloisio. 2025. MANILA Documentation. https://manila-fairness.github.io/
[15] Giordano d'Aloisio. 2025. MANILA Github Repository. https://github.com/giordanoDaloisio/manila-web
[16] Giordano d'Aloisio. 2025. MANILA Video Demonstration. https://youtu.be/f2aXkk56DfY
[17] Giordano d'Aloisio, Antinisca Di Marco, and Giovanni Stilo. 2023. Democratizing Quality-Based Machine Learning Development through Extended Feature Models. In *Fundamental Approaches to Software Engineering*, Leen Lambers and Sebastián Uchitel (Eds.). Springer Nature Switzerland Cham, Springer Nature Switzerland, Cham, 88–110.
[18] Giordano d'Aloisio, Claudio Di Sipio, Antinisca Di Marco, and Davide Di Ruscio. 2025. How fair are we? From conceptualization to automated assessment of fairness definitions. *Software and Systems Modeling* (2025), 1–27.
[19] Giordano d'Aloisio, Andrea D'Angelo, Antinisca Di Marco, et al. 2023. Debiaser for Multiple Variables to enhance fairness in classification tasks. *Information Processing & Management* 60, 2 (2023), 103226. https://doi.org/10.1016/j.ipm.2022.103226
[20] Wirth F. Ferger. 1931. The Nature and Use of the Harmonic Mean. *J. Amer. Statist. Assoc.* 26, 173 (1931), 36 – 40. https://doi.org/10.1080/01621459.1931.10503148
[21] Ioannis Giagkiozis and Peter J. Fleming. 2014. Pareto Front Estimation for Decision Making. *Evolutionary Computation* 22, 4 (2014), 651–678. https://doi.org/10.1162/EVCO_a_00128 arXiv:https://direct.mit.edu/evco/article-pdf/22/4/651/1530439/evco_a_00128.pdf
[22] P. M. Goncalves Jr. and R. S. M. Barros. 2011. Automating Data Preprocessing with DMPML and KDDML. In *2011 10th IEEE/ACIS International Conference on Computer and Information Science*. 97–103. https://doi.org/10.1109/ICIS.2011.23
[23] Valerio Grossi, Beatrice Rapisarda, Fosca Giannotti, et al. 2018. Data science at SoBigData: the European research infrastructure for social mining and big data analytics. *International Journal of Data Science and Analytics* 6, 3 (2018), 205–216. https://doi.org/10.1007/s41060-018-0126-x
[24] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems* 212 (2021), 106622. https://doi.org/10.1016/j.knosys.2020.106622
[25] Brittany Johnson and Yuriy Brun. 2022. Fairkit-learn: a fairness evaluation and comparison toolkit. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings (ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 70–74. https://doi.org/10.1145/3510454.3516830
[26] Michelle Seng Ah Lee and Jat Singh. 2021. The Landscape and Gaps in Open Source Fairness Toolkits. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3411764.3445261
[27] PM Lerman. 1980. Fitting segmented regression models by grid search. *Journal of the Royal Statistical Society Series C: Applied Statistics* 29, 1 (1980), 77–84.
[28] Ji Liu, Esther Pacitti, Patrick Valduriez, et al. 2015. A survey of data-intensive scientific workflow management. *Journal of Grid Computing* 13, 4 (2015), 457–493.
[29] Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, et al. 2022. Software Engineering for AI-Based Systems: A Survey. *ACM Transactions on Software Engineering and Methodology* 31, 2 (2022), 37e:1–37e:59. https://doi.org/10.1145/3487043
[30] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, et al. 2021. A Survey on Bias and Fairness in Machine Learning. *ACM Comput. Surv.* 54, 6, Article 115 (2021), 35 pages. https://doi.org/10.1145/3457607
[31] Henry Muccini and Karthik Vaidhyanathan. 2021. Software Architecture for ML-based Systems: What Exists and What Lies Ahead. In *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)*. 121–128. https://doi.org/10.1109/wain52551.2021.00026
[32] Thanh Nguyen, Maria Teresa Baldassarre, Luiz Fernando de Lima, and Ronnie de Souza Santos. 2024. From Literature to Practice: Exploring Fairness Testing Tools for the Software Industry Adoption. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '24)*. Association for Computing Machinery, New York, NY, USA, 549–555. https://doi.org/10.1145/3674805.3695404
[33] ONU. [n. d.]. ONU Sustainable Development Goals. https://www.un.org/sustainabledevelopment/
[34] PalletsProject. 2023. Jinja website. https://jinja.palletsprojects.com/
[35] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[36] Per Runeson, Martin Host, Austen Rainer, et al. 2012. *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons.
[37] Mauno Rönkkö, Jani Heikkinen, Ville Kotovirta, et al. 2015. Automated preprocessing of environmental data. *Future Generation Computer Systems* 45 (2015), 13–24. https://doi.org/10.1016/j.future.2014.10.011
[38] Pedro Saleiro, Benedict Kuester, Loren Hinkson, et al. 2018. Aequitas: A bias and fairness audit toolkit. *arXiv preprint arXiv:1811.05577* (2018). http://arxiv.org/abs/1811.05577
[39] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2019. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 56–65.
[40] Alfa Yohannis and Dimitris Kolovos. 2022. Towards Model-Based Bias Mitigation in Machine Learning. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems* (Montreal, Quebec, Canada) *(Models '22)*. Association for Computing Machinery, New York, NY, USA, 143–153. https://doi.org/10.1145/3550355.3552401