Fairness Set and Forgotten: Mining Fairness Toolkit Usage in Open-Source Machine Learning Projects

Alfonso Cannavale, Gianmario Voria, Antonio Scognamiglio, Giammaria Giordano, Gemma Catolino, Fabio Palomba

Software Engineering (SeSa) Lab - Department of Computer Science, University of Salerno, Italy

Abstract

Context. The development of machine learning (ML) systems in high-stakes domains has amplified concerns about fairness, prompting the creation of fairness toolkits offering metrics and mitigation techniques. Open-source software (OSS) ecosystems, a critical driver of AI innovation, present a unique opportunity to study the practical adoption of these toolkits. **Objective.** This paper aims to empirically characterize the adoption of fairness toolkits in OSS ML projects by investigating *for what purposes* they are used and *how* their usage evolves over time. **Methods.** We conducted a mining study on GitHub repositories related to real-world ML projects that integrate fairness toolkits such as AIF360 and Fairlearn. Starting from $1,096$ candidate repositories, we applied systematic filtering to identify a final dataset of 20 relevant ML projects (comprising $5,777$ total commits). We analyzed toolkit usage by examining invoked APIs and commit histories to uncover patterns of adoption and evolution. **Results.** Our findings reveal that fairness toolkits are predominantly used for diagnostic purposes, with analytic components integrated early in the project lifecycle and rarely modified thereafter. In contrast, mitigation techniques are infrequently adopted, tend to appear later, and exhibit short, unstable lifespans. **Conclusion.** Our results show that the adoption of fairness toolkits in OSS ML projects is limited and often restricted to initial diagnostic phases, with active mitigation practices remaining rare. These findings highlight the need for improved support to foster more sustained and effective integration of fairness practices within open-source development.

Keywords: Software Engineering for Artificial Intelligence; Machine Learning Fairness Engineering; Fairness Toolkits; Mining Software Repository Study.

1. Introduction

The development of machine learning (ML) systems demands particular attention to non-functional requirements, which are often difficult to quantify but critically influence social sustainability, i.e., the ability of technologies to promote equitable, inclusive, and long-term societal

Email addresses: `acannavale@unisa.it` (Alfonso Cannavale), `gvoria@unisa.it` (Gianmario Voria), `a.scognamiglio32@studenti.unisa.it` (Antonio Scognamiglio), `giagiordano@unisa.it` (Giammaria Giordano), `gcatolino@unisa.it` (Gemma Catolino), `fpalomba@unisa.it` (Fabio Palomba)

well-being [13, 33, 60]. Among these requirements, fairness has emerged as a particularly pressing concern, especially as ML systems are increasingly deployed in high-stakes domains such as criminal justice, healthcare, finance, and education [34, 38, 39, 54]. In these settings, biased models risk reinforcing structural inequalities and disproportionately impacting historically marginalized groups [40, 43]. Ensuring fairness, therefore, is not only a technical challenge but also a necessary safeguard against the ethical and societal risks posed by algorithmic decision-making. Unlike properties such as accuracy or latency, fairness is fundamentally an ethical concept tied to legal norms, social values, and context-specific interpretations, which makes it particularly difficult to operationalize within technical workflows [35].

In response to these challenges, both the artificial intelligence (AI) and software engineering (SE) communities have developed a wide range of technical approaches aimed at measuring and mitigating unfairness. These include analytical **fairness metrics** to assess bias [32] and various **bias mitigation techniques** operating at different stages of the ML pipeline (e.g., pre-, in-, and post-processing) [18, 27, 61]. While the technical effectiveness [10, 25] and practical adoption challenges [15, 24, 31, 45] of these solutions have been increasingly studied, investigations have largely centered on industrial or controlled settings. This leaves a gap in understanding how these fairness interventions are utilized within the dynamic and decentralized context of open-source software (OSS) development.

The practical relevance of fairness is further remarked by the significant efforts of practitioners and tool vendors in creating analytical toolkits that implement the research approaches proposed so far [31], with well-known examples like IBM's AIF360 [4] and Microsoft's Fairlearn [5]. Yet, despite these substantial efforts, *it remains important to ask whether they have effectively translated into practical adoption.* Preliminary research has investigated the adoption of fairness toolkits by analyzing developers' perspectives through qualitative methods, including surveys and interviews conducted in structured industrial settings [15, 24, 31, 45]. While these studies offer valuable insights into organizational barriers and usability challenges, they primarily reflect contexts characterized by formalized processes, managerial oversight, and institutional support. In contrast, *the adoption and use of fairness toolkits in open-source software (OSS) development remains largely unexplored.* This gap is particularly significant because **OSS is not just an alternative development environment, but also plays an important role in the development of AI-enabled systems**. Foundational machine learning libraries and frameworks such as TensorFlow [16], PyTorch [28], and HuggingFace Transformers [58] are developed and maintained within OSS communities. These ecosystems serve not only as technical infrastructure, but also as dynamic spaces for innovation and experimentation, where new AI tools and practices, including fairness-enhancing techniques, can diffuse rapidly [6]. Indeed, open-source ecosystems provide access to a wide and diverse set of AI-enabled systems, ranging from foundational machine learning frameworks to applied solutions across multiple domains [12, 21]: as a proof of that, the GitHub's 2024 Octoverse report[1] highlights that over 70,000 new public AI projects were created in 2024 alone, marking a 98% year-over-year growth and bringing the total to nearly 150,000 such projects. The critical contribution of OSS in accelerating AI innovation makes it a particularly important context for analyzing the adoption of fairness toolkits. It offers a concrete opportunity to assess whether the efforts of researchers and tool vendors have effectively translated into practical impact. In addition, analyzing the adoption of fairness toolkits in OSS is particularly compelling given the distinct characteristics of these communities. Unlike industrial

[1] GitHub's 2024 Octoverse report: `https://github.blog/news-insights/octoverse/octoverse-2024`

environments previously studied, OSS communities are shaped by decentralized governance, voluntary contributions, and evolving workflows—factors that may pose distinct challenges for integrating and sustaining fairness practices [19]. These dynamics may create unique challenges for integrating fairness practices, raising important questions about the alignment between existing toolkit designs and the realities of community-driven development. Furthermore, OSS ecosystems offer a concrete opportunity to assess the practical adoption of fairness toolkits by mining publicly available software repositories, enabling empirical observation of how fairness-related tools and practices are incorporated (or not) into real-world development workflows.

Building on these observations, this paper takes a first step toward empirically characterizing the use of fairness toolkits in open-source ML projects. Specifically, we investigate for *what purposes* these toolkits are adopted and *how* their use evolves over time. By doing so, we provide concrete evidence of how fairness is addressed in practice, offering insight into developers' levels of awareness, prioritization, and engagement with fairness concerns. Our findings can help identify common usage patterns and barriers to adoption, inform the design of more effective fairness-oriented tools, and establish a baseline for tracking progress toward responsible AI development in open-source communities. Additionally, this work may raise awareness among contributors and practitioners, highlighting areas where additional guidance or intervention is needed. More specifically, the main objective of the study is:

> ◎ **Research Objective.** *Our objective is to investigate the purpose and evolution of fairness toolkits in open-source software development communities.*

To achieve this objective, we conducted a mining study of GitHub repositories related to the development of real-world ML projects that implement fairness toolkits. We began by systematically cataloging the fairness-related API identifiers, i.e., functions, classes, and methods, offered by major toolkits like AIF360 [4] and Fairlearn [5]. Each identifier was manually classified based on whether it supports *analytics* (i.e., fairness measurement) or *solutions* (i.e., bias mitigation techniques). We then searched for these API identifiers within GitHub repositories by analyzing import statements, initially retrieving 1,096 candidate repositories. Prior to analysis, we refined this set through manual filtering designed to isolate practical ML applications. This process resulted in a final dataset of 20 relevant OSS projects (comprising a total of 5,777 commits overall), **representing the current state-of-the-art in detecting the adoption of these fairness toolkits in applied open-source ML development**.

To understand how these toolkits are employed in practice, we first investigated the purpose behind their adoption by analyzing which types of toolkit endpoints are most frequently used and by interpreting developers' intentions as expressed in commit messages. Finally, we explored longitudinal data from commit histories to trace the evolution of fairness toolkit usage in open-source repositories. Our analysis reveals several key findings regarding the practical adoption of fairness toolkits in open-source ML projects. First (**RQ**$_1$ - Purpose), we observe an imbalance: developers overwhelmingly utilize toolkit functionalities for analyzing or measuring fairness (analytics), primarily invoking a narrow set of metric calculation APIs during initial setup or experimentation phases. Conversely, the adoption of algorithmic solutions for bias mitigation is rare, suggesting these techniques are not standard practice. Second (**RQ**$_2$ - Evolution), these two types of functionalities exhibit distinct evolutionary patterns. Analytic components, once introduced (often early in the project), tend to persist for long periods with minimal modification, following a *set-and-forget* model. Mitigation components, when they appear at all, are typically introduced much later, have significantly shorter lifespans, and are rarely modified, indicating potentially experimental or short-term usage rather than sustained integration. These findings suggest that while awareness of fairness measurement exists, the active practice of bias mitigation and continuous fairness monitoring remains underdeveloped in the studied open-source contexts.

To summarize, our research provides the following major contributions:

1. A mining study to gather empirical evidence of the purpose and evolution of fairness toolkits in OSS, highlighting the *set-and-forget* pattern for analytics versus the transient nature of mitigation solutions;
2. A systematic classification of active ML projects on GitHub implementing *'Analytics'* and *'Solutions'* methods available in fairness toolkits to support future research on the adoption of these instruments;
3. An online appendix providing all data and scripts to replicate and verify our study [8].

2. Background and Related Work

We first provide the background underpinning our research. We then discuss related work, positioning and motivating our study within the broader context of existing research efforts.

Background. In decision-making contexts, fairness is commonly defined as the absence of prejudice or favoritism toward individuals or groups based on inherent or acquired characteristics [33, 49]. While fair decision-making is a cornerstone of society, human judgment is often subject to cognitive biases that can lead to discriminatory outcomes [17, 37]. ML systems, when developed without consideration for fairness, can replicate and even amplify existing societal biases [7]. In fact, numerous real-world incidents have demonstrated the tangible consequences of algorithmic bias [29, 36, 50, 55].

In response to these growing concerns, the SE research community, particularly within the domain of Software Engineering for Artificial Intelligence (SE4AI), has developed and proposed a set of metrics, automated techniques, and supporting tools to help practitioners measure and manage fairness concerns throughout the software development lifecycle. These methods are typically operationalized in **fairness toolkits**, which are frameworks and libraries developed by open-source communities or organizations that incorporate a wide range of fairness-related functionalities to support practitioners in developing more fair ML models [31]. Examples of these toolkits are the well-known *FairLearn* [5], designed by Microsoft Research and now maintained as a community-driven open-source project, and *AIF360* [4], developed by IBM.

Fairness toolkits usually provide two different types of functionalities, which are (1) metrics or visualization tools to *analyze* bias, and (2) algorithmic *solutions* to mitigate bias.

Verma et al. [52] pioneered the research on fairness measurement, dividing 20 fairness metrics into five groups based on the theoretical definitions. Majumder et al. [32] demonstrated that many metrics are semantically similar, measuring the same aspect of the data. Furthermore, they pointed out that most existing fairness criteria are mutually incompatible. They clustered 26 classification metrics into seven groups and four dataset metrics into three groups, with each group measuring different things against another.

Algorithmic solutions are typically classified into three approaches: pre-processing, in-processing, and post-processing. Pre-processing methods adjust training data to reduce bias before model training. An example is the Reweighting technique, provided by the AIF360 toolkit

[4], that adjusts instance weights to enhance fairness [27]. In-processing techniques modify learning algorithms to mitigate bias during training. For instance, Zhang et al. [61] used adversarial learning to balance fairness during the training phase of the model, and it is implemented in various toolkits such as Fairlearn [5]. Finally, post-processing methods operate by adjusting model outputs to improve fairness without altering the training process. Toolkits such as Themis [18] and Aequitas [51] provide access to these approaches.

Several empirical studies have investigated the effectiveness of bias mitigation techniques provided by fairness toolkits. These works, such as those leveraging the Fairea benchmarking framework [10, 25], have systematically evaluated mitigation strategies across different algorithms and sensitive attributes [9, 13, 62]. While these investigations offer valuable insights—highlighting, for instance, the trade-offs between fairness and accuracy—their scope has largely been limited to controlled research environments. As a result, the real-world usage and adoption of these techniques in software projects remains underexplored.

Related Work and Motivation. While prior work has thoroughly examined the misalignment between fairness toolkits and practitioner needs, much of it has centered on perceived usefulness, usability challenges, or organizational barriers, rather than on actual usage in the wild. For instance, Lee and Singh [31] identified gaps between the functionality of open-source fairness toolkits and the expectations of ML practitioners through interviews, focus groups, and surveys. Similarly, Holstein et al.[24] and Deng et al.[15] explored industry-facing challenges in developing fair ML systems, reporting limitations in toolkit usability and integration based on surveys and observational studies. The main findings of these investigations highlight important obstacles to fairness adoption. However, they rely primarily on self-reported experiences or controlled engagements, which may not reflect how these toolkits are used, or whether they are used at all, in real-world ML development workflows.

Beyond technical limitations, structural and organizational dynamics further complicate the adoption of fairness practices. For example, Rakova et al. [45] examined how internal culture, team structures, and leadership influence responsible ML practices. Likewise, Ayling et al. [2] showed that fairness-related accountability mechanisms are typically inward-facing, with limited transparency or external evaluation. Furthermore, Voria et al. [53] have analyzed why software practitioners adopt fairness toolkits in the industry through technology acceptance theory, highlighting the key role of managerial interventions in fostering adoption.

Taken together, the discussed literature highlights that while fairness toolkits have attracted substantial research attention, particularly regarding their design, benchmarking, and perceived utility in formal organizational settings [31], their use in open-source ML development remains largely unexplored. Prior studies have focused on industry environments, not considering the unique dynamics of open-source communities, where development is decentralized, volunteer-driven, and organically structured [19]. Given the central role of OSS in advancing ML innovation [48], and the availability of publicly accessible repositories, this context offers a unique opportunity to empirically assess whether fairness toolkits are being adopted and how well they align with real-world development practices.

Motivation and Contribution.

Despite the growing availability of fairness toolkits and industry-oriented evaluations, we still lack empirical understanding of how these tools are adopted in open-source ML development, an increasingly influential yet understudied domain. Our study addresses this gap by systematically mining open-source repositories to investigate the purposes for which fairness toolkits are employed and how their usage evolves over time. In doing so, we aim to provide evidence to support both toolkit designers and researchers in understanding how fairness practices emerge and persist in decentralized, community-driven environments.

3. Research Design

The *goal* of this study is to empirically examine how fairness toolkits are adopted and used in open-source machine learning development. The *purpose* is to shed light on whether, how, and why fairness interventions make their way into real-world ML repositories beyond corporate or academic environments, particularly within open-source communities. The study addresses the *perspective* of both researchers, who design and evaluate fairness toolkits, and practitioners, who contribute to and maintain open-source ML projects. By mapping actual usage patterns, identifying common motivations, and highlighting gaps between intended design and practical adoption, we aim to provide actionable insights that foster the adoption of fairness toolkits and promote responsible ML practices.

Fairness toolkits have become a central proposal in responsible AI, offering developers practical means to audit and mitigate bias. Yet, beyond their theoretical appeal and the increasing number of available libraries [31], it remains unclear how often and in what contexts these tools are actually adopted. To understand the practical relevance and application of these toolkits within open-source ML projects, it is crucial to uncover *how* they are used and for *which purpose*. While many toolkits offer both diagnostic tools (e.g., bias metrics and visualizations) and algorithmic solutions for mitigation [4, 5], it is unclear whether developers engage with these toolkits only to assess fairness or to address unfairness actively.

Examining this distinction is critical for understanding the practical interest of developers in the current landscape of ML development, complementing the findings of existing research in industrial settings [15]. Investigating usage patterns can help determine whether fairness concerns are integrated throughout the ML workflow or if toolkit usage is limited to surface-level checks. Furthermore, our objective is to shed light on the intention behind the adoption of fairness toolkits by developers, corroborating existing research [53] from an open-source perspective. Hence, we defined our first research question.

> **RQ$_1$ - Purpose**
>
> *For which purpose are fairness toolkits used in open-source repositories?*

Beyond the purpose of usage, it is important to examine how toolkit integration evolves over time within open-source projects. Analyzing whether fairness-related components are persistently maintained, updated, or only introduced temporarily provides insight into whether fairness is treated as a sustained concern or a one-time evaluation. Assessing adaptation in response to iterative development reflects engagement with fairness objectives. To evaluate this, we defined our second research question.

> **RQ$_2$ - Evolution**
>
> *How do fairness toolkits evolve in open-source repositories?*

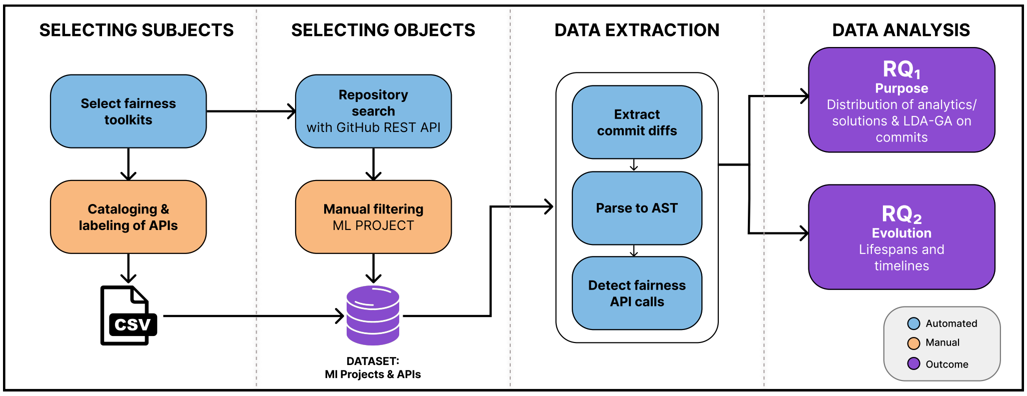
SELECTING SUBJECTS
SELECTING OBJECTS
DATA EXTRACTION
DATA ANALYSIS
Select fairness toolkits
Cataloging & labeling of APIs
CSV
Repository search with GitHub REST API
Manual filtering ML PROJECT
DATASET: Ml Projects & APIs
Extract commit diffs
Parse to AST
Detect fairness API calls
RQ₁ Purpose Distribution of analytics/ solutions & LDA-GA on commits
RQ₂ Evolution Lifespans and timelines
Automated
Manual
Outcome

Figure 1: An overview of our research method. The process consists of four main phases: *Selecting Subjects*, *Selecting Objects*, *Data Extraction*, *Data Analysis*. Blue blocks denote automated processes, orange blocks represent manual validation steps, and purple blocks indicate final data artifacts.

Figure 1 provides an overview of our research method. As shown, we first identified the *subjects* of the study, namely existing fairness toolkits, through an automated search and manual refinement. Subsequently, we selected the *objects* of analysis, i.e., real-world GitHub repositories that depend on these toolkits, applying a combination of dependency analysis and manual filtering. The resulting dataset underwent a structured **data extraction** phase, where commit diffs were parsed into abstract syntax trees (ASTs), fairness-related API calls were detected, and temporal usage metrics (lifespans) were computed. These artefacts form the empirical basis for the **data analysis** phase, through which we systematically address **RQ$_1$** (toolkit usage purpose) and **RQ$_2$** (evolution over time). In Figure 1, blue blocks denote automated processes, orange blocks highlight manual validation steps, and purple blocks represent intermediate or final outputs.

Our study follows empirical research standards, adhering to the guidelines of Wohlin et al. [57] and *ACM/SIGSOFT Empirical Standards* [46],[2] specifically aligned with *"General Standard"* and *"Repository Mining Standard"* due to the nature of our investigation.

3.1. Subjects and Objects of the Empirical Study

This section describes the process used to select the *subjects* and *objects* of our empirical investigation. First, we describe the criteria and procedure used to identify a representative set of fairness toolkits and their relevant APIs (the *subject* of usage analysis). Second, we outline the process adopted to extract the GitHub repositories and filter them to obtain our final dataset of ML projects (the *object* of the study) that expose APIs of these toolkits (Figure 1).

3.1.1. Selecting Fairness Toolkits

To investigate the adoption of fairness toolkits in open-source projects, our first step was to identify a set of relevant toolkits for our mining approach. To achieve this, we leveraged literature that explores the current landscape of open-source ML fairness toolkits [15, 31]. These studies collectively survey and discuss several prominent toolkits, including AIF360, Fairlearn, Aequitas, Themis-ML, FairML, Pymetrics Audit-AI, Google What-if tool, and scikit-fairness.

[2] Available at: `https://github.com/acmsigsoft/EmpiricalStandards`.

However, to ensure that the selected toolkits were compatible with our mining approach and suitable for analysis, we defined four inclusion criteria as shown in Table 1.

Table 1: The four inclusion criteria used to select fairness toolkits for our mining study.

Criterion	Description	Motivation
1. Python Implementation	The toolkit must be primarily implemented in Python.	Most of the ML open-source projects are primarily written in Python.
2. Programmatic API for Workflow Integration	The toolkit must provide a reliable and well-documented Application Programming Interface (API)—including functions, classes, and methods—specifically designed for developers to import and call programmatically within their Python scripts or notebooks.	Our study measures adoption by detecting these specific code interactions. This criterion excludes tools primarily designed for standalone, interactive graphical user interfaces without a primary focus on code-level integration.
3. Fairness-Specific API	The toolkit must provide a clearly identifiable set of APIs dedicated to fairness tasks (measurement or mitigation). Its usage, particularly through import statements, should be reasonably distinguishable from general-purpose ML libraries during static analysis.	This criterion favors libraries with distinct namespaces or modules explicitly focused on fairness.
4. Open-Source Availability	The toolkit's source code must be publicly accessible to allow for verification and broader understanding.	Open-source projects can be analyzed by reading without copyright restrictions.

The systematic application of these criteria led to the exclusion of certain toolkits mentioned in the literature. More particularly, we excluded the following:

- Google What-if Tool: this was excluded as it primarily functions as an interactive visualization tool (often within TensorBoard) and does not offer a conventional programmatic API for direct integration into typical ML code pipelines - it violates Criterion #2.

- scikit-fairness / scikit-lego: it was excluded because, while offering useful functionalities, it acts more as an extension library for scikit-learn. Its API usage can be difficult to reliably distinguish from standard scikit-learn calls via static analysis, potentially impacting the accuracy of our mining results (it violates Criterion #3).

In contrast, all the other toolkits coming from the analysis of the related literature [15, 31] satisfied our inclusion criteria. Hence, we included AIF360, Fairlearn, Aequitas, Themis-ML, FairML, and Audit-AI. Furthermore, we realized that the literature approached in this stage did not explicitly refer to the TensorFlow Model Remediation (TFMR) library,[3] likely because it gained visibility after the surveys were published (in 2022). Given TensorFlow's central role in the ML community, we conducted a targeted manual review of TFMR. Our analysis confirmed that TFMR meets all technical inclusion criteria: it is Python-based, exposes a programmatic API (`tensorflow_model_remediation`), focuses on fairness, and is open-source. To ensure broader ecosystem coverage and account for emerging tools, we therefore included TFMR in our final set of selected fairness toolkits, despite its absence from the initial literature review.

Table 2 reports the final set of toolkits selected for our study.

[3] https://www.tensorflow.org/responsible_ai/model_remediation?hl=en

Table 2: The final set of fairness toolkits selected for our study after applying the inclusion criteria.

Toolkit	Developer/Main Origin	Key Features/Focus
AIF360 [4]	IBM Research	Comprehensive metrics, mitigation algorithms (pre-, in-, post-processing).
Fairlearn [5]	Microsoft Research	Group fairness metrics, mitigation algorithms (post-processing, reduction-based).
Themis-ML [3]	Community/Research	Focus on discrimination measurement (e.g., causality).
Aequitas [47]	UChicago (DSaPP)	Bias and fairness auditing, report generation, practical use cases.
Audit-AI [44]	Pymetrics	Toolkit for auditing ML models, similar focus to Aequitas.
FairML [1]	Community/Research	Emphasis on explainability for fairness, feature importance for bias.
TensorFlow Model Remediation [22]	Google	Libraries for fairness in TensorFlow ecosystem.

3.1.2. Identifying & Categorizing Toolkits APIs

To empirically study the *adoption* and *purpose* of these toolkits, it was necessary to go beyond the toolkit level to identify the specific APIs that developers import and invoke. These APIs represent the *units of analysis* of our quantitative usage analysis. Consequently, compiling a comprehensive list of API endpoints for each selected toolkit was a necessary prerequisite before we could systematically develop and implement the automated procedure for detecting their usage in GitHub repositories. This API identification was first conducted by two authors of this paper, both of whom have more than five years of experience in software engineering and ML development, and was later reviewed by all the other authors until a joint agreement was reached. They conducted a thorough examination of the official documentation for each selected toolkit, including user guides, tutorials, API references, and code examples.

The primary goal of this examination was to extract all identifiable API identifiers (such as classes, methods, functions, and other callable objects) that developers can directly import and invoke to perform fairness-related tasks. Furthermore, to enable a deeper analysis of how these toolkits are used, each extracted API identifier was categorized based on its primary function. More specifically, we established two mutually exclusive functional groups:

- **Analytics** identifiers: these APIs are designed to assess, measure, visualize, or diagnose fairness and bias without actively modifying the underlying data, model training process, or predictions. Their purpose is therefore limited to *observation* and *diagnosis*, primarily satisfying the *analytical* needs of developers. Examples include functions to calculate fairness metrics (e.g., *demographic_parity_difference*) or plot bias visualizations.

- **Solutions** identifiers: these APIs implement algorithms or techniques that actively intervene to mitigate identified bias. Their purpose is therefore *interventional*, aiming to address fairness concerns. From a practical perspective, this includes modifying the data (pre-processing, e.g., *Reweighing*), altering the model learning process (in-processing, e.g., *ExponentiatedGradient*), or adjusting the model's outputs (post-processing, e.g., *CalibratedEqOddsPostprocessing*).

This distinction between observation and diagnosis (*Analytics*) and active intervention and mitigation (*Solutions*) encompasses the spectrum of functionalities offered by the fairness-related APIs within the selected toolkits. An API either calculates or visualizes a fairness aspect or actively attempts to alter the system towards a fairer state. Therefore, each identified API identifier was assigned to exactly one of these two categories.

To ensure inter-rater reliability for this classification task, the first two authors of this paper, who both have extensive experience in fairness engineering and had previously engaged with the use of the analyzed toolkits, worked in close collaboration. They jointly examined the documentation and code examples for each extracted API to determine its primary intended purpose. Ambiguous cases regarding functionality or classification were discussed immediately to reach a consensus. After this initial collaborative phase, the preliminary categorization was presented and discussed during five one-hour meetings with all the remaining authors. We decided to set a maximum time limit of one hour for these sessions in order to mitigate potential biases due to fatigue, loss of attention, or variations in concentration. In these sessions, each contested or unclear case was collectively reviewed, and the rationale followed by the two primary annotators was examined, refined when needed, and formally confirmed by the entire author team. The resulting comprehensive list of classified API identifiers provided the ground truth for the subsequent automated search and is available in the online appendix [8].

3.1.3. Selecting and Filtering Repositories: Defining the Objects of Study

After cataloging all fairness-related functionalities provided by the toolkits, we developed an automated procedure to systematically search GitHub repositories that potentially use these toolkits. Using the GitHub REST API, we conducted targeted searches within Python source files (`.py`) and Jupyter notebooks (`.ipynb`) for import statements referencing the components of the identified toolkit. The analysis excluded commented code and verified the actual presence of import statements in active source code to minimize false positives.

This phase was conducted in March 2025 and yielded an initial dataset of $1,096$ candidate repositories. However, preliminary inspection revealed significant heterogeneity, with many repositories unrelated to practical ML development (e.g., educational materials, replication packages, and small experiments). Analyzing this entire set could bias findings about the actual adoption of fairness toolkits by practitioners in sustained development contexts. Therefore, to ensure our analysis accurately targets relevant practices and to mitigate this potential threat, we performed a manual classification of all $1,096$ repositories.

The objective was to categorize repositories based on their primary function and purpose, allowing us to isolate those representing our main objects of study: *'ML Project'*, defined as repositories focused on the practical application or development of a specific ML model. To this end, two authors independently classified each repository by applying the categorization scheme detailed in Table 3. The classification was based on the inspection of multiple artifacts of the repositories, including README files, directory structures, code comments, and example usages, to determine whether a repository corresponded to an *'ML Project'* or belonged to other types such as *'Fairness Tool'*, *'ML Tool'*, *'ML Toy Project'*, *'Academic/Educational Repository'*, or *'Replication Package of a Research Article'*. Any classification discrepancies were resolved through discussion and double-checking.

The manual classification revealed that the vast majority of repositories fall into categories unrelated to direct application in ML development. Specifically, 25.1% are replication packages (i.e., projects reproducing experiments) and 23.6% are academic repositories (containing course

Table 3: The categorization scheme applied during the manual classification.

Category	Description
Fairness Tool	Toolkits, libraries, frameworks, or platforms that implement fairness techniques (mitigation, measurement, auditing, etc.)
ML Tool	General-purpose tools for developing, training, or managing AI models. Includes libraries, frameworks, toolkits, or management systems
ML Project	Repositories focused on the practical application or development of a specific AI model
ML Toy Project	Small-scale, experimental, or personal projects, often for learning or testing specific functionalities
Academic/Educational	Code for demos, tutorials, or university experiments not directly linked to a published paper
Replication Package	Code created for the reproduction of experiments from scientific papers
Fork of Fairness Tool	Forks of known fairness toolkits
Unclassified	Repositories with missing/non-English READMEs or whose purpose could not be determined
Other	Repositories not clearly fitting into the preceding categories

materials, tutorials, etc.). These two categories alone account for nearly half of the dataset, highlighting the strong presence of fairness toolkits primarily in research and educational contexts, rather than in applied ML projects. Other categories include repositories implementing fairness tools themselves (10.5%), general-purpose ML tools (7.7%), and ML toy projects (2.3%), with a small fraction being forks of fairness toolkits (1.4%) (detailed in Figure 2).

As such, only 31 repositories (2.8% of the initial $1,096$) were classified as potentially meeting the criteria for *'ML Project'*. To ensure the quality and relevance of this subset for analyzing practical adoption, we performed a final filtering and deduplication process on these 31 candidates, which consisted of the following actions: As such, only 31 repositories (2.8% of the initial $1,096$) were classified as potentially meeting the criteria for *'ML Project'*. To ensure the quality and relevance of this subset for analyzing practical adoption, we performed a final filtering and deduplication process, consisting of the following actions:

- We excluded direct forks of projects already selected (6 forks of kserve/kserve), as they do not represent independent adoption efforts.

- We removed repositories containing largely identical code to another project already included (one instance: Stock-CloseCast) to avoid redundancy in the analysis.

- In cases where a single project used multiple toolkits, we corrected the dataset to count the repository only once (this happened in one case: Healthcare PoC), hence preserving an accurate project-level granularity.

- We excluded repositories where toolkit imports appeared only in non-executable documentation or illustrative code snippets (2 instances: AI-Driven-Anomaly-Detection-for-Mitigating-Quantum-Computing-Attacks and Optimizing-Patient-Selection-for-Primary-Prevention-ICD-Implantation), as they did not reflect practical usage.

After applying these filtering steps, we obtained the final dataset comprising 20 distinct *'ML Project'* repositories. This subset (Table 4) forms the core focus of our subsequent investigation into the purpose and evolution of fairness toolkit usage. All data pertaining to the classification

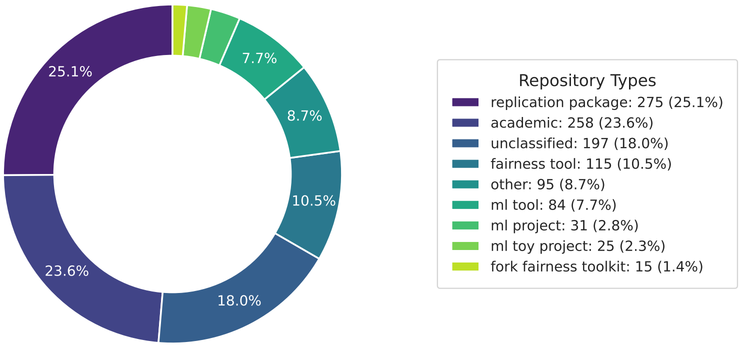

Figure 2: Distribution of the $1,096$ candidate repositories across the categories defined in Table 3.

and filtering stages are available in our online appendix [8]. Table 4 provides details for each repository included in this final dataset.

Looking at Table 4, the final set of 20 ML projects exhibits heterogeneity across various dimensions. Project popularity varies dramatically, from highly starred repositories like kserve/kserve (over $4,000$ stars) to many with zero or few stars. Similarly, development activity, proxied by the number of commits, ranges from established projects with over $1,700$ commits (kserve/kserve) to projects with very limited commit histories (e.g., fewer than 10 commits for Healthcare_PoC, AirQualityPM25, and content-recommendation). This diversity reflects the inclusion of projects at different stages of maturity and levels of community engagement, all selected according to our manual classification criteria (Section 3.1.3). While some repositories exhibit limited historical activity, their inclusion still provides valuable insights into fairness toolkit adoption across different stages of project maturity. Analyzing projects at early development phases allows us to observe whether and how fairness considerations are integrated from the outset, and whether adoption patterns differ compared to more established projects. Early-stage or smaller-scale projects may rely on fairness toolkits differently, for instance, focusing more on experimentation or initial diagnostics rather than long-term bias mitigation. Including a diverse set of projects ensures that our analysis captures a broader range of adoption behaviors, avoiding a bias toward only mature, highly active repositories. The subsequent analysis (Section 3.3) investigates how fairness toolkits are used across this heterogeneous landscape.

Beyond the analysis of the 20 ML Projects presented in this paper, our manual classification of the initial $1,096$ candidate repositories represents a contribution in itself. This comprehensive dataset, categorizing repositories according to the scheme in Table 3 (e.g., *ML Project*, *Academic/Educational*, *Replication Package*, *Fairness Tool*), provides a resource for the research

Table 4: The final dataset of 20 ML Projects selected for in-depth analysis, with their key metadata and application context.

Repository	#Stars	#Commits	First Commit	Last Commit	#LOC	Context
Denbergvanthijs/HCML	0	63	2023-06-08	2023-07-03	1,617	Finance
HaeghJulie/PBL-HGO	0	268	2022-12-27	2023-06-05	9,828	Healthcare
MLOPS-IE7374-Fall2024-G9/mlops-project	1	756	2024-09-19	2024-12-10	9,504	Energy Forecasting
hiaac-finance/credit_pipeline	0	245	2023-06-13	2025-02-05	7,418	Finance
IE7374-MachineLearningOperations/StockPricePrediction	2	169	2024-10-17	2024-12-10	7,998	Finance
madhurima-vanga/Amazon-Customer-Sentiment-Analyser	0	113	2024-09-30	2024-12-17	3,859	E-commerce
KARSarma/Air-Quality	2	603	2024-09-30	2024-12-09	2,513,466	Environmental
mrudulaacharya/Parkinson-s-Prediction	2	414	2024-10-31	2024-12-09	4,604	Healthcare
Seun-Ajayi/Tourmaline	0	44	2022-09-03	2023-11-01	726	Demographic
alecpanattoni/MissingnessFairnessAnalysis	0	58	2023-01-08	2023-03-15	1,295	Analysis
AyoubMaimmadi/Audieyes	0	147	2024-02-25	2024-05-27	9,445	Healthcare
rajpandeygithub/Automated-BiLingual-Complaint-System	1	354	2024-09-30	2024-12-09	6,851	Finance
kserve/kserve	4,099	1,771	2019-03-27	2025-04-23	217,505	MLOps Platform
theyorubayesian/cliffhanger	1	43	2021-06-09	2021-11-24	604	Demographic
charutapaliwal/Healthcare_PoC	0	3	2024-10-06	2024-10-06	1,544	Healthcare
kanagalasrilakshmi/AirQualityPM25	0	8	2024-12-01	2024-12-01	6,369	Environmental
Surjeet2110/content-recommendation	0	7	2024-07-13	2024-07-13	290	Social
Venkata1106/Stock_Price_Prediction_MLOPS	0	95	2024-11-28	2024-12-09	5,126	Finance
Pranavbp525/PersonalTrainerAI	0	288	2025-01-28	2025-04-24	24,357	Healthcare
KodeJaiSurya/Vibify	2	328	2024-09-28	2024-12-10	2,710	Entertainment

community. It enables further investigations into various facets of fairness toolkit adoption and usage across different segments of the open-source ecosystem, potentially focusing on contexts beyond applied ML development, such as educational trends or the structure of replication studies. The complete classified dataset is available in our online appendix [8] to facilitate such replication and extension studies.

3.2. Data Extraction

After selecting the ML project repositories, we systematically analyzed their use of fairness toolkits. Our analysis focused exclusively on the default branch of each repository—typically `main` or `master`—as configured on GitHub, to ensure we captured the primary line of development. We adopted an automated, static code analysis approach that examined both the complete commit history and the most recent state of each default branch. To trace the evolution of toolkit usage over time, we analyzed every commit involving changes to Python source files (`.py`) or Jupyter Notebooks (`.ipynb`), the most common formats in ML development [20, 42, 56]. For each of these changes, we applied code parsing techniques to detect the introduction, removal, or modification of calls to the previously identified fairness-related components.

For the parsing step, we built on top of similar previous work [11, 23]. In-depth, for `.py` files, we parsed the Python source code into its corresponding Abstract Syntax Tree (AST) structure. This tree representation enables a precise analysis of the code's syntax, distinguishing function and method calls from other language elements, such as variables or comments. We traversed the AST, identifying nodes that represent function or method invocations (i.e. `call` nodes). For `.ipynb`, the process first involved programmatically extracting the executable Python code contained within the notebook's cells. This extracted code was then parsed into an AST, and the same call detection logic used for `.py` files was applied.

To detect the introduction or removal of toolkit calls, this AST-based approach compared the code structure before and after each commit. The comparison of AST between an interval of commits is not something new; indeed, it is used especially to identify refactoring operations [59]. Specifically, by comparing the ASTs generated from the pre- and post-commit versions of a modified file, we identified `call` nodes representing function or method invocations that were either removed (present only in the pre-commit AST) or added (present only in the post-commit AST). We then checked if the name of the function or method being invoked in these added or removed calls matched any identifier present in our pre-compiled list of fairness toolkit APIs. In other terms, this *differential AST analysis* ensures that we capture actual invocation changes rather than mere mentions in comments or strings.

Our AST traversal logic was designed to capture two common invocation patterns: direct function or class instantiations (e.g., `AdversarialDebiasing()`) and attribute-based method calls (e.g., `metrics.ratio()`, where `ratio` is the detected identifier). However, we clarify that our static analysis captures only explicit API references and does not resolve aliases or dynamic invocations. For instance, an invocation like `AD()` after an import statement `from aif360 import AdversarialDebiasing as AD` would not be detected. This conservative design choice prioritizes precision over recall, ensuring that all detected calls are unambiguously attributable to a known fairness toolkit API. While this approach might under-report the absolute number of usages, it provides a reliable and precise lower bound on toolkit adoption.

For each instance of a fairness identifier involved in such a detected within a historical commit, we extracted a structured set of metadata, including: commit information (identifier, date), details of the modified file (path), the specific fairness identifier involved, its functional classification (*'Analytics'* or *'Solutions'*) and library of origin, and the type of change observed (e.g., *'Call Added'* or *'Call Removed'*). Commits that modified files containing fairness API calls without altering the set of detected fairness invocations were not recorded as distinct change events for the purpose of this API usage tracing. In addition to this structural data, which reflects the widespread use of commit messages for understanding software evolution, we extracted the associated commit message for every commit identified as relevant in this phase (i.e., a commit that introduces or modifies the usage of a fairness identifier).

While the commit history analysis tracks changes over time ($\mathbf{RQ}_2$), it does not provide a complete inventory of all fairness toolkit usages present in the final version of the code. To obtain this comprehensive snapshot of the current integration state ($\mathbf{RQ}_1$), we conducted a dedicated analysis of the latest commit (*snapshot*) of each repository. This involved parsing all `.py` and `.ipynb` files within the project's final state to identify all current usages of fairness identifiers, recording: the path of the file containing the usage, the specific identifier detected, its classification (*'Analytics'* or *'Solutions'*), and the library of origin.

The combined output of these two analyses—the evolutionary track from commit history and the final state inventory of the latest snapshot—constitutes the raw dataset, which contains point-in-time occurrences and the evolution of fairness toolkit usage within the selected projects. This dataset was subsequently used for the *Data Analysis* phase, described in Section 3.3.

3.3. Data Analysis

To address our two research questions, we analyzed the data extracted from the identified open-source ML repositories (Table 4). This phase involved examining both the current state of fairness toolkit integration and the motivations behind their usage over time. We employed a *mixed-methods* approach combining quantitative analysis of identifier usage with qualitative examination of commit messages. All the code used to perform the analyses and the data collected are available in our online appendix [8].

RQ$_1$ — Purpose. To address **RQ**$_1$, we first performed a quantitative analysis of the latest code snapshot. We focused on the distribution and frequency of API calls classified as *'Analytics'* versus *'Solutions'* across the 20 ML projects and the different toolkit libraries. This allowed us to characterize the primary purpose (measurement versus mitigation) of fairness toolkit integration in the current state of these projects. To gain deeper insights into the developers' intentions behind integrating or modifying these fairness features, we complementarily analyzed the relevant commit messages using Topic Modeling. Following a text preprocessing stage designed to clean, standardize, and focus the messages on meaningful terms (removing noise such as stopwords and programming keywords) as indicated by Kozanidis et al. [30], we proceeded with the analysis of commit messages. Then, we applied the Latent Dirichlet Allocation (LDA) algorithm [26]. We optimized the LDA model's parameters to maximize the coherence of the identified topics, employing a genetic algorithm-based variant known as LDA-GA [41].

The interpretation of the resulting topics involved a qualitative analysis process conducted independently by first two authors of the paper. They examined the top representative keywords generated by the LDA model for each identified topic to discern their semantic focus. Based primarily on these keywords, the authors collaboratively assigned a concise and meaningful thematic label that summarizes the core concept or activity represented (e.g., *'Model Update'*, *'Initial Setup'*). Any initial disagreements between the two annotators were resolved through discussion to reach a shared consensus. Finally, the resulting topic labels were reviewed and validated by the remaining authors to confirm their coherence and consistency. This keyword-driven interpretation helped us uncover recurring themes and gain insights into the developers' rationale for integrating or modifying fairness-related code.

RQ$_2$ — Evolution. To address **RQ**$_2$, we analyzed the longitudinal data derived from the commit history. For each detected usage event (addition or removal) of a specific fairness identifier invocation within a repository, we calculated its usage lifespan. This lifespan represents the time elapsed between the commit where the invocation was first added and the commit where it was last observed as removed (i.e., no longer detected in the post-commit AST compared to the pre-commit AST). If an invocation, once added, was never detected as removed throughout the analyzed history, its lifespan extended to the date of the last analyzed commit. Additionally, we counted the number of intermediate commits that modified the file containing the specific identifier invocation during its calculated lifespan, regardless of whether the modification affected the fairness API call itself. Performing this analysis separately for *'Analytics'* and *'Solutions'* identifiers allowed us to compare their evolutionary dynamics. By examining the distributions of these lifespans and intermediate commit counts, we assessed usage persistence (long-term vs. transient) and the level of development activity for the integration.

Furthermore, to visualize the overall temporal presence and activity per toolkit library, we generated timeline charts. Each chart displays horizontal bars per library, spanning from the earliest first usage date to the latest last usage date observed for any identifier of the library across all projects. Overlaid on these bars are markers indicating the timestamps of the intermediate commits that modified files containing APIs from that library.

4. Analysis of the Results

This section presents the results of our study for each of the two defined research questions.

Table 5: Frequency of fairness module invocations in the final code snapshot of the 20 ML Projects, sorted by count.

Module	Count	Type
fairlearn.metrics	**50**	**Analytics**
aif360.sklearn.metrics	**27**	**Analytics**
aif360.metrics	**7**	**Analytics**
src.aequitas.group	**4**	**Analytics**
themis_ml.metrics	**4**	**Analytics**
aif360.sklearn.inprocessing	3	Solutions
src.aequitas.bias	**3**	**Analytics**
aif360.sklearn.postprocessing	2	Solutions
src.aequitas.fairness	**2**	**Analytics**
src.aequitas.plotting	**2**	**Analytics**
aif360.algorithms.postprocessing	1	Solutions
fairlearn.postprocessing	1	Solutions

*4.1. **RQ$_1$** — Purpose*

To understand *why* developers rely on fairness toolkits, we combined a static analysis of the latest code snapshot with a semantic evaluation of commit histories. First, our analysis of the latest snapshot reveals a pronounced disparity in how these toolkits are used: among the 106 fairness API invocations identified across the 20 ML projects, 99 (93.4%) target analytics components, such as computing metrics or visualizing bias issues, while only 7 calls (6.6%) invoke bias mitigation algorithms. In other words, developers overwhelmingly use fairness toolkits to measure or expose bias, rather than to correct it.

This pattern is reinforced when we examine the distribution of API calls at the **module** level. Two metric modules dominate usage: `fairlearn.metrics` from FairLearn [5] accounts for 50 calls and `aif360.sklearn.metrics` from AIF360 [4] for 27 (detailed in Table 5). Combined, these two modules alone account for 77 invocations, representing 72.6% of all 106 fairness API calls identified in our dataset. By contrast, modules that implement mitigation strategies are rarely used. For example, `aif360.sklearn.inprocessing` from AIF360 [4] appears only three times, and all other solution modules are referenced once or twice at most. Hence, we conclude that the current use of fairness toolkits is mainly diagnostic rather than interventional.

To better understand developers' intent when introducing or modifying fairness-related code, we focused on the 128 commits identified through our historical analysis (Section 3.2) that specifically introduced, removed, or otherwise altered the usage of fairness toolkit APIs. We analyzed the messages of these commits using topic modeling (LDA-GA) [41]. Unlike manual coding based on a predefined codebook, LDA-GA is an unsupervised algorithm that discovers latent thematic structures (topics) from the textual data by identifying co-occurring patterns of words. While the model surfaced 17 latent topics, four emerged as clearly dominant, collectively accounting for 76% of the commit messages, as shown in Figure 3. These topics represent common usage patterns:

- *'Update model/feature'* (35 commits): fairness checks are preserved or adjusted as part of broader model or pipeline updates. For example, a commit with the message *"Update #Feature selection and model predictions"* falls under this theme.

- *'Add file/upload'* (32): toolkits are introduced at the setup stage, often in notebooks, de-

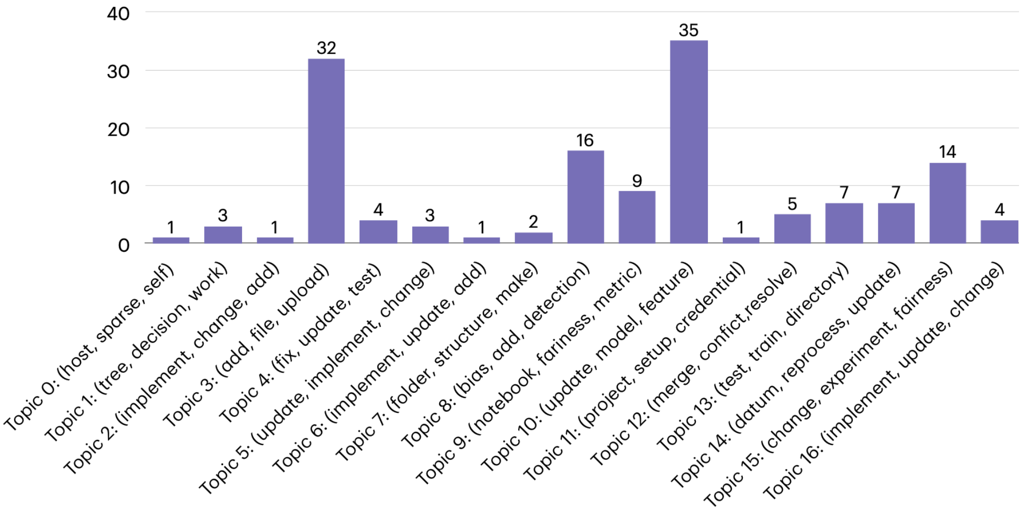

Figure 3: Distribution of the 128 commit messages across the 17 latent topics identified by LDA-GA.

pendency files, or CI scripts. A representative commit message for this topic is simply *"Add files via upload"*.

- *'Bias detection added'* (16): metric calls are added to audit models for fairness issues. Commit messages such as *"added bias detection code"* and *"Add support for bias detection using AIF360"*.
- *'Change experiment fairness'* (14): fairness metrics are used in comparative experiments or ablation studies. A commit with the message *"changes on experiments, to run only a subset of fairness models"* illustrates this usage pattern.

Together, these findings suggest that fairness tools are primarily employed to support model development and experimentation activities. Linguistic cues from the commit messages further support this interpretation: terms explicitly referencing mitigation efforts (e.g., "debias", "reweight", "repair") appear in fewer than 5% of the commit logs.

☞ Answer to RQ$_1$.

In open-source ML projects, fairness toolkits serve almost exclusively as analytic dashboards: developers import a narrow set of metric APIs and invoke them when setting up, experimenting with, or modifying a model, while bias-mitigation algorithms remain largely untouched in everyday practice.

4.2. *RQ$_2$ — Evolution*

Analyzing the commit history allowed us to investigate the evolution of fairness toolkit usage over time, revealing how different types of fairness interventions are integrated and maintained within the lifecycle of open-source ML projects. Our findings point to two contrasting evolutionary profiles for analytic and mitigation functionalities, suggesting different levels of integration

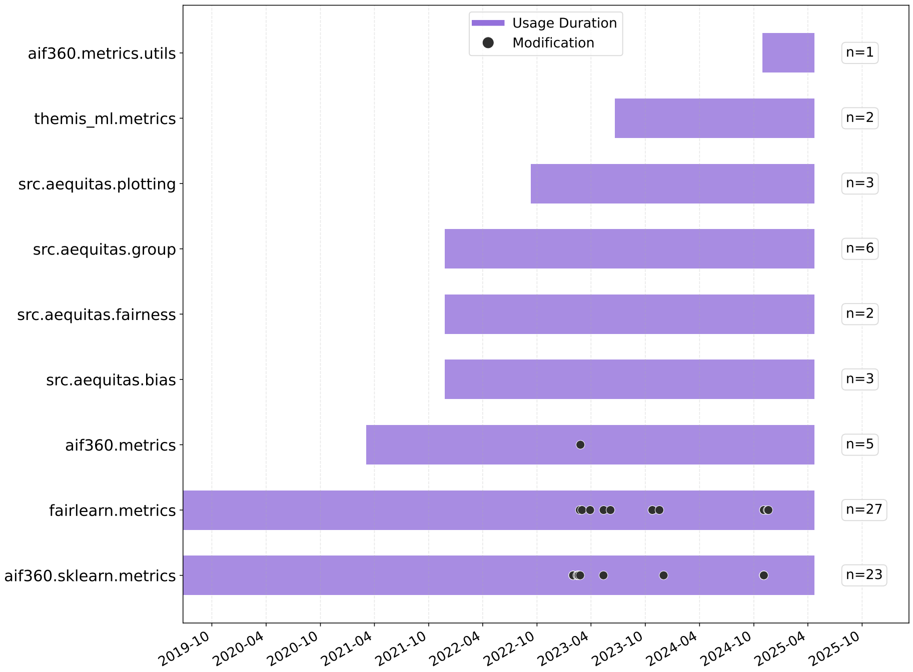

Figure 4: Usage timelines for *Analytic* (metrics) modules across the 20 ML projects.

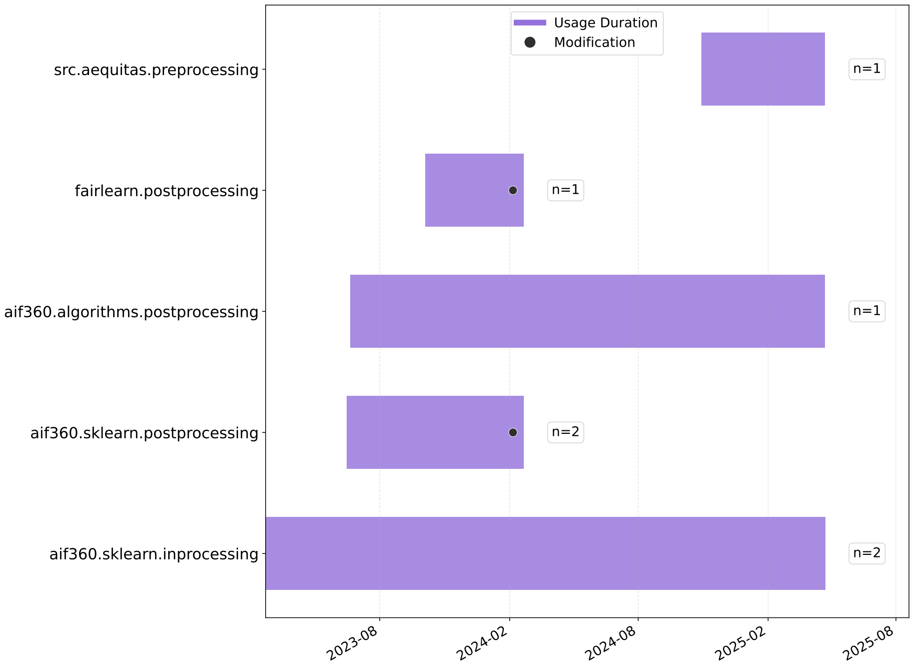

Figure 5: Usage timelines for *Solution* (mitigation) modules across the 20 ML projects.

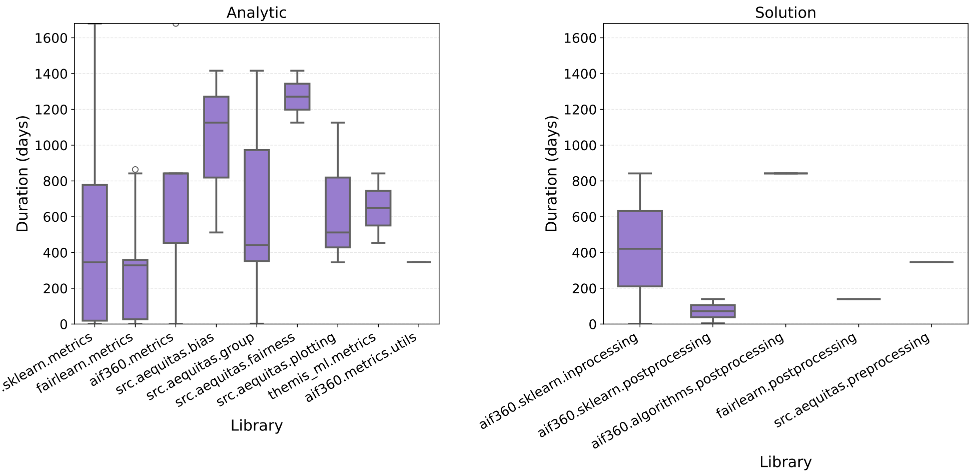

Figure 6: Boxplot comparison of usage lifespans for *Analytic* and *Solution* modules.

maturity and developer engagement. These evolutionary dynamics also align with our findings for **RQ**$_1$, where the focus on analytic APIs in the code snapshot reflects the persistent usage of measurement tools, contrasting with the rarity of mitigation API calls, which corresponds to their often short-lived integration observed here.

Analytics live long and quiet. Core metrics (`fairlearn.metrics`, `aif360.sklearn.metrics`) are introduced early—often with the first commit of a project—and remain active for 3–4 years (*median* ≈ 620 days, Figure 6). Figure 4 shows no red dots (indicating modifications), meaning the import statement often survives unchanged throughout the observation period. When a modification occurs, it is typically a minor update, potentially in a notebook ($n \leq 4$ modification commits per library). Removals are rare: only two analytic identifiers disappear during our observation window. This persistence suggests that once integrated, fairness measurement becomes a stable, albeit passive, component of the development workflow, possibly fulfilling reporting or basic auditing needs without requiring frequent adjustments. The minimal churn might indicate either stable requirements regarding fairness metrics or perhaps a lack of deeper engagement with evolving fairness definitions or toolkit capabilities.

Solutions are late, short-lived, and almost immutable. Mitigation packages appear one to two years after the corresponding metric modules and were found in only four projects. Their median lifetime is 140 days, and they are virtually untouched after the initial import (Figure 5 shows at most one red dot per bar). Two solution identifiers were removed after less than a month. The contrast with analytics is quantified in Figure 6: analytic lifetimes are an order of magnitude longer ($IQR \approx$ 350–900 days) than solution lifetimes ($IQR \approx$ 50–180 days). This pattern suggests that bias mitigation is not yet a mature or integrated practice in these open-source projects. The late introduction, short lifespan, and near-absence of post-import modifications for mitigation APIs sharply contrast with the profile of analytic APIs. This suggests that active mitigation is often explored experimentally or temporarily, rather than being adopted as a sustained practice, potentially due to challenges like technical complexity or performance trade-offs.

☞ Answer to RQ$_2$.

In open-source ML projects, fairness toolkits follow a *set-and-forget* pattern: metric APIs (Analytics) are added early and persist with minimal churn, indicating stable integration for measurement purposes. Mitigation algorithms (Solutions), conversely, appear late, are often short-lived, and rarely modified, suggesting experimental usage rather than sustained deployment. Consequently, continuous fairness monitoring seems limited to passive measurement, lacking active, evolving debiasing practices within the observed project lifecycles.

5. Discussion and Take-Away Messages

The results of our study reported a number of findings that are worth noting and that lead to actionable implications for researchers and practitioners.

5.1. On the Adoption of Fairness Toolkits

First and foremost, our findings reveal that **fairness toolkits are rarely adopted in open-source ML projects**. Even when fairness toolkits are adopted, our analysis shows that mitigation efforts tend to occur late in the development timeline and are often short-lived. This pattern suggests that **fairness interventions are frequently treated as optional additions rather than integral components of ML workflows**. This trend was evident already during the preliminary dataset construction phase, where only a very small percentage of repositories showed active use of fairness functionalities. This limited adoption represents a key result of our study, suggesting two non-exclusive interpretations: first, that practitioners may lack awareness of the availability and relevance of fairness toolkits; second, that significant practical barriers to adoption persist, including usability challenges, poor integration with existing development workflows, and perceived overhead. Regardless of the underlying cause, **the data point to a clear gap between the development of fairness-enhancing technologies and their real-world uptake**. This finding aligns with prior research that has identified similar challenges in the adoption of fairness toolkits within industrial contexts, where organizational barriers, usability issues, and misalignment with practitioner needs were reported as key obstacles [15, 24, 31, 45].

The disparity between the adoption of analytic and mitigation components could be explained by several factors. First, mitigation techniques are inherently more intrusive than diagnostic metrics. They actively alter the ML pipeline—by modifying data (pre-processing), the learning algorithm (in-processing), or model predictions (post-processing)—which can negatively impact model performance (e.g., accuracy) or introduce unintended side effects. Second, there may be a poor fit with the practitioners' existing technology stack and workflows. The need to integrate a new, specialized library may conflict with established development processes, especially if it requires significant changes to the language, framework, or infrastructure. Finally, practitioners might prefer implementing custom fairness checks using general-purpose tools they are already familiar with, rather than adopting a dedicated fairness toolkit. This approach could offer them more control and better integration, even if it means forgoing the standardized, specialized functionalities offered by fairness toolkits.

These findings represent a call to action for researchers and toolkit designers. Researchers may want to investigate further the *socio-technical factors influencing fairness adoption*, going beyond purely technical evaluations of fairness metrics or mitigation algorithms. In particular, future work could explore how fairness concerns emerge (or fail to emerge) during different phases of ML project development, what organizational or community dynamics support sustained fairness practices, and how developer workflows could be better aligned with fairness goals. Longitudinal field studies, developer interviews, and mixed-method investigations could shed light on the real-world socio-technical frictions that current tools fail to address.

Toolkit designers may want to prioritize *Human-Computer Interaction principles to improve usability and integration*. This includes simplifying API designs (e.g., providing one-line decorators for fairness enforcement), offering easy-to-adopt default configurations for common use cases, embedding fairness checks into familiar development environments such as Jupyter notebooks or IDEs, and providing ready-to-use integrations with CI/CD pipelines. In other terms, toolkits should not be isolated libraries requiring extra effort, but rather should be presented as lightweight, low-friction extensions of the developer's existing workflow. By combining both technical effectiveness and socio-technical integration, future fairness toolkits can move from being optional add-ons to becoming standard practice in responsible ML development.

Take-Away Message 1. Researchers should investigate socio-technical barriers to fairness adoption, while toolkit designers should focus on improving usability, integration, and developer experience to embed fairness practices into standard ML workflows.

5.2. Bridging the Transparency and Relevance Gap in Fairness Adoption

As a follow-up discussion point, our study reveals not only the scarcity of fairness toolkit adoption within open-source ML development, but also **a striking lack of transparency even when fairness functionalities are present**. Out of the initial 1,096 repositories, only 2.8% involved fairness toolkit imports within ML projects, narrowing to 20 unique projects after filtering. While toolkit usage appears more common in academic and replication settings, its integration into practical, applied ML development remains exceptional. More concerningly, our analysis of commit messages (**RQ**$_1$) showed that **fairness-related actions were rarely accompanied by explanatory notes**. Commit messages typically referenced model updates or initial setup, but seldom documented the rationale for choosing specific fairness metrics, thresholds, or mitigation strategies. This opacity hinders future contributors, reviewers, or even the original developers from understanding whether fairness was genuinely considered or how fairness interventions were evaluated. This dual challenge, i.e., minimal adoption combined with poor documentation, highlights a critical misalignment between academic approaches to fairness engineering and real-world practitioner behavior. While research has produced a wealth of fairness metrics and mitigation techniques, these innovations are often perceived as peripheral and lack the transparency necessary for collective learning and accountability.

Addressing this disconnect requires action at multiple levels. OSS developers and communities should not only consider adopting fairness toolkits but also actively document fairness-related decisions within codebases—through commit messages, code comments, and project documentation. Toolkit designers could facilitate this process by integrating structured documentation templates, automatic fairness reporting tools, or visual dashboards directly into toolkits. Meanwhile, researchers should investigate why fairness remains a peripheral concern in practice, and work to realign fairness tooling efforts with developers' real constraints, workflows, and priorities—again, this requires a *socio-technical* connotation of the problem.

Take-Away Message 2. Developers and OSS communities should improve the transparency of fairness-related practices by documenting fairness decisions, rationales, and mitigation actions within project artefacts. Researchers should investigate the socio-technical barriers behind limited fairness adoption and work toward aligning fairness tooling with practitioners' workflows, priorities, and needs.

5.3. Toward Continuous Fairness Monitoring and Mitigation

Our findings reveal that fairness analytics APIs tend to be introduced early in a project's lifecycle and remain integrated with minimal modification. At the same time, bias mitigation efforts are rare and often short-lived. This *set-and-forget* pattern raises significant concerns: while developers recognize the value of diagnostic fairness measurements, they frequently **treat fairness as a static property validated once, rather than as a dynamic quality requiring ongoing maintenance**. In dynamic environments, fairness is susceptible to *decay* or *drift* over time, caused by changing data distributions, evolving societal contexts, model updates, or regulatory shifts [14]. Relying solely on initial fairness checks creates a false sense of security, potentially allowing harmful biases to re-emerge unnoticed during a system's lifecycle.

Our results call for a paradigm shift from static fairness checks to continuous fairness monitoring and mitigation. Practitioners should complement initial fairness evaluations with mechanisms for ongoing tracking of fairness metrics, integrating fairness monitoring into continuous integration/continuous deployment workflows. Automated alerts, fairness dashboards, and retraining triggers could help detect and address fairness decay early, minimizing manual overhead and promoting sustainable fairness maintenance. Researchers and tool designers, in turn, should develop practical solutions that operationalize continuous fairness monitoring. This could include libraries that automatically log fairness metrics over time, visualize fairness trends, and generate pull requests when fairness regressions are detected. Integrating such capabilities into popular MLOps platforms like GitHub Actions, Jenkins, or Azure Pipelines would make fairness monitoring a seamless extension of standard DevOps practices. In other terms, by embedding fairness into operational workflows, developers can move beyond one-off fairness validations and adopt a more resilient, long-term approach to fairness in ML systems.

Take-Away Message 3. Practitioners should move from static fairness checks to continuous fairness monitoring, embedding fairness evaluations into CI/CD workflows. Researchers and toolkit designers should develop automated tools for tracking fairness metrics over time, visualizing fairness trends, and triggering remediation actions when fairness degradation is detected.

6. Threats to Validity

This section discusses potential threats to the validity of our empirical study and the strategies implemented to mitigate them.

Internal Validity. These threats refer to how well the results can be attributed to the interventions or variables studied rather than other factors or confounding variables.

The selection of the 20 ML projects, although based on a systematic classification process (Section 3.1.3), might represent a potential threat. Projects were chosen based on the detectable usage of specific fairness toolkits. Although this was necessary for the purpose of the study, characteristics specific to these 20 projects (e.g., maturity, domain, and development practices beyond the use of the fairness toolkit) could potentially confound the observed results regarding the toolkit's purpose (**RQ**$_1$) and evolution (**RQ**$_2$). We mitigate this by analyzing a diverse set of projects in various domains (Table 4), but we recognize that unobserved project-specific factors may influence our findings. When analyzing the evolution (**RQ**$_2$), external events or internal project shifts (e.g., changes in team composition, major refactorings, shifts in project goals) that occur concurrently with changes in fairness toolkit usage may influence persistence or removal patterns. Our analysis focuses on the observable lifespan and modification history, but does not control for all potential concurrent events that may explain the change in usage.

External Validity. External validity refers to the extent to which our findings can be generalized beyond the specific research context. Our findings are intrinsically dependent on the set of fairness toolkits selected for analysis (Table 2). While we aimed for a representative set based on literature and technical criteria [15, 31] (Section 3.1.1), the exclusion of other toolkits (e.g., those without programmatic APIs or non-Python based) means our results regarding adoption patterns may not generalize to the entire fairness toolkit landscape. The specific functionalities and usability of the included toolkits likely influence their adoption and usage patterns. In addition, the study focused on Python-based ML projects hosted on GitHub, identified via specific import statements. Findings may not generalize to: (i) projects using different programming languages or ML frameworks; (ii) projects hosted on other platforms; (iii) projects that address fairness without using the specific toolkits we searched for (e.g., through custom implementations).

Finally, we acknowledge that the final dataset of 20 ML projects represents a small sample of the OSS ecosystem, which inherently limits the external validity and generalizability of our findings. Although this small number is a finding in itself—reflecting the rarity of practical toolkit adoption—the patterns observed within this group may not be representative of all OSS ML projects that use fairness toolkits. To mitigate this threat and foster a cumulative scientific process, we have made all of our research artifacts, including the list of 1,096 classified repositories, the final dataset of 20 projects, and all analysis scripts, publicly available in our online appendix [8]. This transparency not only ensures the replicability of our study but also facilitates future research. Other researchers can build upon our work by expanding the dataset with more projects, toolkits, or platforms, thereby systematically testing and extending the generalizability of our initial findings.

Construct Validity. These threats relate to the potential discrepancies between the theoretical framework and the actual observations. The accuracy of our study relies on the correct identification and classification (*'Analytics'* vs *'Solutions'*) of fairness toolkit APIs (Section 3.1.2). While performed carefully by two authors examining documentation, potential misclassifications or overlooking relevant APIs could impact the results for **RQ**$_1$ and **RQ**$_2$. We provide the list of classified APIs in the online appendix [8] for transparency.

Our automated data extraction (Section 3.2) uses differential AST analysis to detect API additions/removals. While robust against simple text mentions, this method might: miss usages invoked indirectly (e.g., through complex wrappers or dynamic calls), and fail on unparseable code, although this is typically rare on default branches. We mitigate this by focusing on direct *Call* nodes in the AST. An empirical evaluation of the precision and recall of our detection method was not performed. Defining a reliable ground truth for such an evaluation would require extensive manual inspection of a large sample of code to identify all possible fairness invocations, including aliased and dynamic ones, which is beyond the scope of this exploratory study. Future work could address this by combining static and dynamic analysis to enable a quantitative precision/recall assessment.

The interpretation of topics for **RQ**$_1$ relies on LDA-GA and subsequent manual labeling by two authors. While LDA-GA [26, 41] helps find coherent topics and a consensus was reached, topic modeling inherently involves a degree of subjectivity in interpreting the meaning behind keyword clusters and assigning thematic labels. The identified themes represent dominant patterns but might not capture all nuances of developer intent.

Calculating lifespan based on the first addition and last removal (or last commit analyzed) provides a proxy for persistence. Still, it does not capture the intensity or nature of usage during that period. A toolkit might persist for years but be invoked only rarely. The count of intermediate modification commits provides some insight into activity, but is still an approximation.

Conclusion Validity. Conclusion validity relates to the reliability of our conclusions. The conclusions drawn in this study rely significantly on descriptive statistics (counts and percentages) and visualizations to identify and characterize usage patterns, reflecting the exploratory nature of our investigation into this understudied area. While inferential statistical tests were not extensively employed (except for LDA parameter optimization [41]), the observed magnitudes of differences (e.g., between *'Analytics'* and *'Solutions'* usage, or their lifespans) provide strong indicative evidence for the reported findings. However, readers should interpret claims of *dominance* or *trends* based on the presented descriptive evidence, acknowledging that formal statistical significance testing for all comparisons was outside the scope of this exploratory work and represents an avenue for future research.

7. Conclusion and Future Work

This paper analyzed the adoption and usage patterns of fairness toolkits within open-source ML projects. Through a mining study of open-source GitHub repositories that analyze API calls and commit histories, we investigated the purpose behind the integration of fairness toolkits and their evolution. We identified and categorized fairness APIs, analyzing their frequency, function (analytics vs. mitigation), and persistence. Compared to their prevalence in academic or organizational contexts, fairness toolkits are poorly adopted in OSS, and their usage overwhelmingly favors diagnostic analytics over active bias mitigation solutions. Furthermore, the evolution reveals a *set-and-forget* pattern: analytic tools persist long-term with minimal change, while mitigation tools appear late, are short-lived, and rarely undergo modification, suggesting a primarily experimental use, and indicating potential disconnect between the available tools and practical development workflows, or underlying challenges in integrating mitigation effectively.

The insights gained set the foundation for our future research agenda. First, we plan to expand the scope of our mining study to include repositories from different platforms (e.g., GitLab, Bitbucket) and additional programming languages, to assess the generalizability of the observed patterns. Second, investigating the relationship between specific project characteristics (e.g., size, domain, team structure, contribution patterns) and the adoption (or non-adoption) of fairness toolkits could reveal crucial contextual factors and barriers. Furthermore, a natural extension of this work would be to investigate the *impact and effectiveness* of the observed mitigation practices. This could involve a more fine-grained, execution-based analysis to examine the causal link between the introduction of a mitigation API and subsequent changes in fairness metric values within the projects' evolution. Finally, developing techniques to infer fairness considerations directly from code changes and project artifacts, complementing explicit toolkit usage and potentially identifying custom fairness implementations, represents a promising avenue to gain a more holistic view of fairness practices in OSS. This also includes improving our detection techniques through, for instance, alias resolution and dynamic analysis to enhance coverage without compromising precision.

Credits

Alfonso Cannavale: Formal analysis, Investigation, Data Curation, Validation, Writing - Original Draft, Visualization. **Gianmario Voria**: Formal analysis, Investigation, Data Curation, Validation, Writing - Original Draft, Visualization. **Antonio Scognamiglio**: Formal analysis, Investigation, Data Curation, Validation, Writing - Original Draft, Visualization. **Giammaria Giordano**: Supervision, Validation, Writing - Review & Editing. **Gemma Catolino**: Supervision, Validation, Writing - Review & Editing. **Fabio Palomba**: Supervision, Validation, Writing - Review & Editing.

Conflict of interest

The authors declare that they have no conflict of interest.

Data Availability

The data collected in the context of this research, along with the scripts and the results of the experiments, are publicly available in our online appendix [8].

Acknowledgement

We acknowledge the use of ChatGPT-4 to ensure linguistic accuracy and enhance the readability of this article. This work was partially supported by the European Union – NextGenerationEU, through the Italian Ministry of University and Research, under the PRIN PNRR 2022 project “FRINGE: context-aware FaiRness engineerING in complex software systEms” (grant no. P2022553SL, CUP: D53D23017340001). Additional support was provided by the project FAIR (PE0000013), funded under the Italian NRRP (National Recovery and Resilience Plan), MUR program, and co-financed by the European Union – NextGenerationEU.

References

[1] Julius A Adebayo et al. 2016. *FairML: ToolBox for diagnosing bias in predictive modeling*. Ph. D. Dissertation. Massachusetts Institute of Technology.

[2] Jacqui Ayling and Adriane Chapman. 2022. Putting AI ethics to work: are the tools fit for purpose? *AI and Ethics* 2 (08 2022). `https://doi.org/10.1007/s43681-021-00084-x`

[3] Niels Bantilan. 2018. Themis-ml: A fairness-aware machine learning interface for end-to-end discrimination discovery and mitigation. *Journal of Technology in Human Services* 36, 1 (2018), 15–30.

[4] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilović, 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.

[5] Sarah Bird, Miro Dudík, Richard Edgar, Brandon Horn, Roman Lutz, Vanessa Milan, Mehrnoosh Sameki, Hanna Wallach, and Kathleen Walker. 2020. Fairlearn: A toolkit for assessing and improving fairness in AI. *Microsoft, Tech. Rep. MSR-TR-2020-32* (2020).

[6] Andrea Bonaccorsi and Cristina Rossi. 2003. Why open source software can succeed. *Research policy* 32, 7 (2003), 1243–1258.

[7] Yuriy Brun and Alexandra Meliou. [n. d.]. Software fairness. In *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*.

[8] Alfonso Cannavale, Gianmario Voria, Antonio Scognamiglio, Giammaria Giordano, Gemma Catolino, and Fabio Palomba. 2025. "Fairness Set and Forgotten: Mining Fairness Toolkit Usage in Open-Source Machine Learning Projects". `https://figshare.com/s/3c0c5bd2f39834d43957`

[9] Zhenpeng Chen, Jie M Zhang, Federica Sarro, and Mark Harman. [n. d.]. Fairness Improvement with Multiple Protected Attributes: How Far Are We?. In *Proceedings of the IEEE/ACM 46th ICSE.*

[10] Zhenpeng Chen, Jie M Zhang, Federica Sarro, and Mark Harman. 2023. A comprehensive empirical study of bias mitigation methods for machine learning classifiers. *ACM Transactions on Software Engineering and Methodology* 32, 4 (2023), 1–30.

[11] YunSeok Choi, JinYeong Bak, CheolWon Na, and Jee-Hyong Lee. 2021. Learning Sequential and Structural Information for Source Code Summarization. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, Online, 2842–2851. `https://doi.org/10.18653/v1/2021.findings-acl.251`

[12] Vincenzo De Martino, Gilberto Recupito, Giammaria Giordano, Filomena Ferrucci, Dario Di Nucci, and Fabio Palomba. 2025. Into the ML-Universe: An improved classification and characterization of machine-learning projects. *Journal of Systems and Software* 230 (2025), 112471. `https://doi.org/10.1016/j.jss.2025.112471`

[13] Vincenzo De Martino, Gianmario Voria, Ciro Troiano, Gemma Catolino, and Fabio Palomba. [n. d.]. Examining the Impact of Bias Mitigation Algorithms on the Sustainability of Ml-Enabled Systems: A Benchmark Study. *Available at SSRN 4966447* ([n. d.]).

[14] Oscar Blessed Deho, Lin Liu, Jiuyong Li, Jixue Liu, Chen Zhan, and Srecko Joksimovic. 2024. When the Past != The Future: Assessing the Impact of Dataset Drift on the Fairness of Learning Analytics Models. *IEEE Transactions on Learning Technologies* 17 (2024), 1007–1020. `https://doi.org/10.1109/TLT.2024.3351352`

[15] Wesley Hanwen Deng, Manish Nagireddy, Michelle Seng Ah Lee, Jatinder Singh, Zhiwei Steven Wu, Kenneth Holstein, and Haiyi Zhu. 2022. Exploring How Machine Learning Practitioners (Try To) Use Fairness Toolkits. In *2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*. ACM. `https://doi.org/10.1145/3531146.3533113`

[16] TensorFlow Developers. 2022. TensorFlow. *Zenodo* (2022).

[17] Itiel E Dror and Peter AF Fraser-Mackenzie. 2008. Cognitive biases in human perception, judgment, and decision making: Bridging theory and the real world. In *Criminal investigative failures*. Routledge, 79–94.

[18] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. [n. d.]. Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th FSE.*

[19] Daniel M. German, Gregorio Robles, Germán Poo-Caamaño, Xin Yang, Hajimu Iida, and Katsuro Inoue. 2018. "Was my contribution fairly reviewed?": a framework to study the perception of fairness in modern code reviews. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. Association for Computing Machinery, New York, NY, USA, 523–534. `https://doi.org/10.1145/3180155.3180217`

[20] Giammaria Giordano, Giusy Annunziata, Andrea De Lucia, Fabio Palomba, et al. 2023. Understanding Developer Practices and Code Smells Diffusion in AI-Enabled Software: A Preliminary Study.. In *IWSM-Mensura*.

[21] Danielle Gonzalez, Thomas Zimmermann, and Nachiappan Nagappan. 2020. The state of the ml-universe: 10 years of artificial intelligence & machine learning software development on github. In *Proceedings of the 17th International conference on mining software repositories*. 431–442.

[22] Google. [n. d.]. "Tensorflow Model Remediation". `https://github.com/tensorflow/model-remediation`

[23] Hassan Bapeer Hassan, Qusay Idrees Sarhan, and Árpád Beszédes. 2024. Evaluating Python Static Code Analysis Tools Using FAIR Principles. *IEEE Access* 12 (2024), 173647–173659. `https://doi.org/10.1109/ACCESS.2024.3503493`

[24] K. Holstein, J.W. Vaughan, III Daumé, H., M. Dudík, and H. Wallach. 2019. Improving fairness in machine learning systems: What do industry practitioners need? *Conference on Human Factors in Computing Systems - Proceedings* (2019). `https://doi.org/10.1145/3290605.3300830`

[25] Max Hort, Jie M. Zhang, Federica Sarro, and Mark Harman. 2021. Fairea: A Model Behaviour Mutation Approach to Benchmarking Bias Mitigation Methods. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Athens, Greece) *(ESEC/FSE 2021)*. Association for Computing Machinery, New York, NY, USA, 994–1006. `https://doi.org/10.1145/`

3468264.3468565

[26] Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. 2019. Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey. *Multimedia tools and applications* 78 (2019), 15169–15211.

[27] Faisal Kamiran and Toon Calders. 2012. Data preprocessing techniques for classification without discrimination. *Knowledge and information systems* (2012).

[28] Nikhil Ketkar, Jojo Moolayil, Nikhil Ketkar, and Jojo Moolayil. 2021. Introduction to pytorch. *Deep learning with python: learn best practices of deep learning models with PyTorch* (2021), 27–91.

[29] Tahsin Alamgir Kheya, Mohamed Reda Bouadjenek, and Sunil Aryal. 2024. The Pursuit of Fairness in Artificial Intelligence Models: A Survey. arXiv:2403.17333 [cs.AI] `https://arxiv.org/abs/2403.17333`

[30] Nicholas Kozanidis, Roberto Verdecchia, and Emitza Guzman. 2022. Asking about Technical Debt: Characteristics and Automatic Identification of Technical Debt Questions on Stack Overflow. In *Proceedings of the 16th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement* (Helsinki, Finland) *(ESEM '22)*. Association for Computing Machinery, New York, NY, USA, 45–56. `https://doi.org/10.1145/3544902.3546245`

[31] 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* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 699, 13 pages. `https://doi.org/10.1145/3411764.3445261`

[32] Suvodeep Majumder, Joymallya Chakraborty, Gina R Bai, Kathryn T Stolee, and Tim Menzies. [n. d.]. Fair enough: Searching for sufficient measures of fairness. *ACM Transactions on Software Engineering and Methodology* ([n. d.]).

[33] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–35.

[34] Claire Cain Miller. 2015. Can an algorithm hire better than a human. *The New York Times* 25 (2015).

[35] Seumas Miller. [n. d.]. Machine learning, ethics and law. *Australasian Journal of Information Systems* ([n. d.]).

[36] ABC News. 2009. *Amazon restores rankings for gay-themed books*. `https://abcnews.go.com/Technology/story?id=7343222&page=1`

[37] Wim De Neys, Oshin Vartanian, and Vinod Goel. 2008. Smarter Than We Think: When Our Brains Detect That We Are Biased. *Psychological Science* 19, 5 (2008), 483–489. `https://doi.org/10.1111/j.1467-9280.2008.02113.x` arXiv:https://doi.org/10.1111/j.1467-9280.2008.02113.x PMID: 18466410.

[38] Jianjun Ni, Yinan Chen, Yan Chen, Jinxiu Zhu, Deena Ali, and Weidong Cao. 2020. A survey on theories and applications for self-driving cars based on deep learning methods. *Applied Sciences* 10, 8 (2020), 2749.

[39] Parmy Olson. 2011. The algorithm that beats your bank manager. *CNN Money March* 15 (2011).

[40] Tiago P Pagano, Rafael B Loureiro, Fernanda VN Lisboa, Rodrigo M Peixoto, Guilherme AS Guimarães, Gustavo OR Cruz, Maira M Araujo, Lucas L Santos, Marco AS Cruz, Ewerton LS Oliveira, et al. [n. d.]. Bias and unfairness in machine learning models: a systematic review on datasets, tools, fairness metrics, and identification and mitigation methods. *Big data and cognitive computing* 7 ([n. d.]).

[41] Annibale Panichella, Bogdan Dit, Rocco Oliveto, Massimilano Di Penta, Denys Poshynanyk, and Andrea De Lucia. 2013. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In *2013 35th International conference on software engineering (ICSE)*. IEEE, 522–531.

[42] Miguel Pérez-Francisco, Selin Aydin, and Horst Lichter. 2024. A Flexible Cell Classification for ML Projects in Jupyter Notebooks. *ArXiv* abs/2403.07562 (2024). `https://api.semanticscholar.org/CorpusID:268363905`

[43] Dana Pessach and Erez Shmueli. [n. d.]. A review on fairness in machine learning. *ACM Computing Surveys (CSUR)* ([n. d.]).

[44] Pymetrics. [n. d.]. "Audit-ai". `https://github.com/pymetrics/audit-ai`

[45] Bogdana Rakova, Jingying Yang, Henriette Cramer, and Rumman Chowdhury. [n. d.]. Where Responsible AI meets Reality: Practitioner Perspectives on Enablers for Shifting Organizational Practices. *Proc. ACM Hum.-Comput. Interact.*, Article 7 (apr [n. d.]), 23 pages. `https://doi.org/10.1145/3449081`

[46] Paul Ralph, Sebastian Baltes, Domenico Bianculli, Yvonne Dittrich, Michael Felderer, Robert Feldt, Antonio Filieri, Carlo Alberto Furia, Daniel Graziotin, Pinjia He, Rashina Hoda, Natalia Juristo, Barbara A. Kitchenham, Romain Robbes, Daniel Méndez, Jefferson Seide Molléri, Diomidis Spinellis, Miroslaw Staron, Klaas-Jan Stol, Damian A. Tamburri, Marco Torchiano, Christoph Treude, Burak Turhan, and Sira Vegas. 2020. ACM SIGSOFT Empirical Standards. *CoRR* abs/2010.03525 (2020). arXiv:2010.03525 `https://arxiv.org/abs/2010.03525`

[47] Pedro Saleiro, Benedict Kuester, Loren Hinkson, Jesse London, Abby Stevens, Ari Anisfeld, Kit T Rodolfa, and Rayid Ghani. 2018. Aequitas: A bias and fairness audit toolkit. *arXiv preprint arXiv:1811.05577* (2018).

[48] SÃ¸ren Sonnenburg, Mikio L Braun, Cheng Soon Ong, Samy Bengio, Leon Bottou, Geoffrey Holmes, Yann LeCun, Klaus-Robert MÃžller, Fernando Pereira, Carl Edward Rasmussen, et al. 2007. The need for open source

software in machine learning. *Journal of Machine Learning Research* 8, Oct (2007), 2443–2466.

[49] Christopher Starke, Janine Baleis, Birte Keller, and Frank Marcinkowski. [n. d.]. Fairness Perceptions of Algorithmic Decision-Making: A Systematic Review of the Empirical Literature. arXiv:2103.12016 [cs.HC]

[50] The New York Times. 2021. *Facebook Apologizes After A.I. Puts 'Primates' Label on Video of Black Men*. https://www.nytimes.com/2021/09/03/technology/facebook-ai-race-primates.html

[51] Sakshi Udeshi, Pryanshu Arora, and Sudipta Chattopadhyay. [n. d.]. Automated directed fairness testing. In *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*.

[52] Sahil Verma and Julia Rubin. [n. d.]. Fairness definitions explained. In *2018 ieee/acm international workshop on software fairness (fairware)*. IEEE.

[53] Gianmario Voria, Stefano Lambiase, Maria Concetta Schiavone, Gemma Catolino, and Fabio Palomba. 2024. From Expectation to Habit: Why Do Software Practitioners Adopt Fairness Toolkits? *arXiv preprint arXiv:2412.13846* (2024).

[54] Pin Wang, En Fan, and Peng Wang. 2021. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern recognition letters* 141 (2021), 61–67.

[55] Mengyi Wei and Zhixuan Zhou. 2022. AI Ethics Issues in Real World: Evidence from AI Incident Database. arXiv:2206.07635 [cs.AI]

[56] Ratnadira Widyasari, Zhou Yang, Ferdian Thung, Sheng Qin Sim, Fiona Wee, Camellia Lok, Jack Phan, Haodi Qi, Constance Tan, Qijin Tay, et al. 2023. Niche: A curated dataset of engineered machine learning projects in python. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 62–66.

[57] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, Anders Wesslén, et al. 2012. *Experimentation in software engineering*. Vol. 236. Springer.

[58] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[59] Hyrum K Wright, Daniel Jasper, Manuel Klimek, Chandler Carruth, and Zhanyong Wan. 2013. Large-scale automated refactoring using ClangMR. In *2013 IEEE International Conference on Software Maintenance*. IEEE, 548–551.

[60] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. 2022. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems* 4 (2022), 795–813.

[61] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. [n. d.]. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*.

[62] Mengdi Zhang and Jun Sun. 2022. Adaptive Fairness Improvement Based on Causality Analysis. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Singapore, Singapore) *(ESEC/FSE 2022)*. Association for Computing Machinery, New York, NY, USA, 6–17. https://doi.org/10.1145/3540250.3549103