FAIRNESS ADEQUACY TEST FOR MACHINE LEARNING SYSTEMS

by

Kehinde Oluwasayo Akinola

December, 2025

Director of Thesis: Srinivasan Madhusudan, PhD

Major Department: Computer Science

As Machine Learning (ML) systems assume a larger role in decisions that affect people's lives, such as who receives a loan, access to healthcare, or early release from prison, ensuring these systems are fair is more important than ever. However, current fairness checks often miss the subtle and complex ways in which bias can appear in algorithms.

This dissertation tackles that gap by proposing a practical framework for testing how well machine learning models meet fairness standards, especially across protected groups. Instead of relying solely on standard performance metrics, the approach combines statistical tools with stress testing techniques to uncover hidden or overlooked biases.

There are four main contributions. First, we introduce a fairness adequacy test using metrics like Equal Opportunity Difference and Equalized Odds to examine disparities in error rates across groups. Second, we apply mutation testing by altering sensitive features such as race or gender to see how model outputs change, helping assess fairness under different conditions. Third, we use permutation methods to simulate edge cases and test how models respond to unusual or extreme inputs. Finally, we validate this approach with real-world case studies in areas like healthcare, finance, and criminal justice, where fairness is especially critical.

By offering a clear and testable way to evaluate fairness, this work aims to support

the development of more trustworthy, accountable, and equitable AI systems

FAIRNESS ADEQUACY TEST FOR MACHINE LEARNING SYSTEMS

A Thesis

Presented to The Faculty of the Department of Computer Science

East Carolina University

In Partial Fulfillment of the Requirements for the Degree

Master of Science in Software Engineering

by

Kehinde Oluwasayo Akinola

December, 2025

FAIRNESS ADEQUACY TEST FOR MACHINE LEARNING SYSTEMS

by

Kehinde Oluwasayo Akinola

APPROVED BY:

DIRECTOR OF THESIS:	Srinivasan Madhusudan, PhD
COMMITTEE MEMBER:	Peng Kebin PhD
COMMITTEE MEMBER:	John Reisch, PhD
CHAIR OF THE DEPARTMENT OF COMPUTER SCIENCE:	Venkat Gudivada, PhD
DEAN OF THE GRADUATE SCHOOL:	Paul J. Gemperline, PhD

Table of Contents

LIST OF TABLES

LIST OF FIGURES

List of Acronyms

AI Artificial Intelligence. x

CART Classification and Regression Trees.. vi, x, 28

EOD Equal Opportunity Difference. x

EOM Equalised Odds Metrics. x

FAT Fairness Adequacy Test. x

ML Machine Learning. x, 1

PCA Principal Component Analysis. x

SPD statistical parity difference. x

Chapter 1

Introduction

1.1 What Is Testing and the Adequacy Principle

A model can report 95% accuracy and still deny loans to a protected group 20% more often. That gap between statistical performance and social impact is why testing remains our first line of defense. In classical software, adequacy asks whether a test suite exercises the paths most likely to reveal defects (29). Machine-learning systems change the question: models learn from historical data, behave stochastically, and operate in social contexts that shift over time (16).

Adequacy therefore moves from code coverage to *slice coverage*: systematic exercise of fairness-relevant subpopulations, feature contexts, and plausible distributional shifts. An adequate fairness test suite should surface fairness-affecting faults, not only accuracy regressions. Following a sociotechnical view (16), we measure impact on individuals and communities with the same rigor as error rates.

We make this operational with mutation testing. We generate small, realistic data mutations, apply distribution and validity checks, retrain the model, and test whether the suite detects a meaningful change in a fairness metric such as Equal Opportunity Difference or Equalized Odds. A detected change "kills" a mutant; otherwise it "survives." We decide kill versus survive using a data-driven threshold estimated from the base model's null variability on the same test suite (mean $+1.5\,\mathrm{std}$); see **??**.

The resulting mutation score summarizes how effectively the suite reveals fairness faults across sensitive slices and realistic shifts.

1.2 Why Fairness Testing Is Necessary

Standard evaluation metrics like accuracy, precision, and recall are useful, but they often fail to reveal subgroup harms that arise from historical bias, proxy labels, or distribution shift (25; 37). Because machine learning now informs decisions about creditworthiness, employment, criminal justice, and health, these hidden failures can scale into systemic discrimination.

Fairness testing treats equity as a first-class, testable property. It detects *fairness bugs*—unwarranted associations between protected attributes (for example gender, race, or age) and model outputs—and it guides targeted repair rather than superficial metric fixes (14; 38). Property-based and property-driven approaches make fairness requirements explicit so models can be checked systematically across subpopulations (37). Survey work likewise argues that stakeholders expect evidence of equitable treatment, not only overall utility (25). Tools such as FairTest and Themis demonstrate practical detection of statistically significant disparities across modalities from tabular data to text and vision (14; 38).

A well-known case underscores the point. Amazon discontinued an experimental recruiting tool after fairness analysis found that résumés with indicators of female gender were systematically downgraded, despite high aggregate performance (11). Targeted fairness testing—not overall accuracy—surfaced the problem.

1.3 Recall of Global Incidents Caused by Inadequate Fairness Testing

High-profile incidents show how insufficient fairness testing leads to concrete harm. Each vignette pairs a missed test signal with an adequacy lesson that our framework makes actionable. First incident reported is related to criminal justice. Investigation

Next incident is realted amaozn protype.

Criminal justice: COMPAS risk assessment (2016). Investigations reported higher false-positive rates for Black defendants at similar rearrest rates to white defendants (2; 7). *Missed signal:* subgroup confusion matrices and uncertainty for Equalized Odds components before adoption. *Adequacy lesson:* evaluate race slices with label-noise stress tests and apply a data-driven kill rule to flag unfair error profiles.

Hiring: Amazon recruiting prototype (2018). A resume screener trained on male-dominated histories down-ranked female-associated terms (11). *Missed signal:* shortlist parity checks and feature-attribution audits during model development. *Adequacy lesson:* balanced gender slices with realistic data mutations would have eliminated variants that encode gender proxies.

Healthcare: population-health risk scoring (2019). A widely used risk model used cost as a proxy for clinical need, under-referring Black patients with equal risk (30). *Missed signal:* label-validity checks and subgroup error analysis under proxy shifts. *Adequacy lesson:* mutate the cost proxy and substitute clinical-need surrogates to expose Equal Opportunity gaps.

Education: UK A-level grading algorithm (2020). Standardization downgraded many students, with disproportionate impact by school type and cohort size; the policy was reversed (33; 36). *Missed signal:* evaluation on small-cohort slices, uncertainty reporting, and distributional stress tests before rollout. *Adequacy lesson:* slice coverage by school type and cohort size, plus mutations of historical distributions, would have flagged disparate impact.

Public benefits: Dutch childcare benefits scandal (2018–2021). Automated risk profiling wrongly flagged thousands of families, disproportionately those with dual nationality; the government resigned in 2021 (32; 18). *Missed signal:* sensitivity to protected attributes and enforcement of proxy bans. *Adequacy lesson:* feature guardrails that ban protected attributes and proxies, combined with nationality-feature mutations, would have surfaced discriminatory associations.

Digital platforms: ad delivery and targeting (2019). Studies showed ad delivery could produce discriminatory reach for jobs, housing, and credit even under targeting restrictions; platforms committed to changes (1; 23). *Missed signal:* end-to-end outcome testing by demographic slice under realistic budget and creative variation. *Adequacy lesson:* mutate creatives and budgets and measure slice-level reach and error metrics to detect disparate delivery.

Media presentation: image-cropping bias (2020–2021). User reports and internal analysis led to product changes after cropping favored certain faces (20). *Missed signal:* visual slice coverage across skin tone and gender in real posting conditions. *Adequacy lesson:* pre-launch tests that vary aspect ratios and face positions, scored by subgroup saliency metrics, would have surfaced bias.

Summary. Across domains, aggregate validation hid subgroup harms. A fairness-adequacy practice with explicit slice coverage, realistic mutation scenarios, and uncertainty-aware kill rules provides evidence that fairness-affecting faults are detected before deployment.

1.4 Implications of Not Conducting Fairness Testing

The incidents in section 1.3 reveal four classes of risk.

Legal and compliance. In regulated domains such as credit, employment, healthcare, insurance, and investing, the absence of auditable fairness testing can trigger investigations, fines, and litigation. Regulators increasingly expect pre-deployment evidence that subgroup effects were evaluated and mitigated.

Institutional trust. Publicized disparities reduce stakeholder confidence and raise the cost of assurance. Teams without fairness testing struggle to provide credible explanations to executives, auditors, and affected users.

Technical debt. Aggregate metrics can mask fairness faults that appear only in specific slices or under realistic shifts. Without slice coverage, distribution checks, and uncertainty-aware decision rules, harmful associations persist and resurface during retraining or domain change.

Social and ethical harm. Unexamined systems can entrench disadvantage in lending, hiring, health, education, and justice. Small disparities compound into lasting inequities.

Practical takeaway. Make fairness a first-class test requirement. Define slice coverage, adopt uncertainty-aware thresholds, report mutation scores with confidence intervals, and maintain artifacts that show fairness-affecting mutants would have been detected before deployment.

1.5 Implications of Not Conducting Fairness Testing

The incidents in section 1.3 reveal four classes of risk.

- **Legal and compliance.** In regulated domains such as credit, employment, healthcare, insurance, and investing, the absence of auditable fairness testing can trigger investigations, fines, and litigation. Regulators increasingly expect pre-deployment evidence that subgroup effects were evaluated and mitigated.
- **Institutional trust.** Publicized disparities reduce stakeholder confidence and raise the cost of assurance. Teams without fairness testing struggle to provide credible explanations to executives, auditors, and affected users.
- **Technical debt.** Aggregate metrics can mask fairness faults that appear only in specific slices or under realistic shifts. Without slice coverage, distribution checks, and uncertainty-aware decision rules, harmful associations persist and resurface during retraining or domain change.
- **Social and ethical harm.** Unexamined systems can entrench disadvantage in lending, hiring, health, education, and justice. Small disparities compound into lasting inequities.

Practical takeaway. Make fairness a first-class test requirement. Define slice coverage, adopt uncertainty-aware thresholds, report mutation scores with confidence intervals, and maintain artifacts that show fairness-affecting mutants would have been detected before deployment.

Table 1.1: Documented Failures Due to Fairness Testing Omissions Across Critical Domains

Domain	**Incident (Year)**	**Outcome**	**Type of Harm**	**Missed Fairness Signal**
Justice	COMPAS bias in sentencing (2016)	Black defendants received higher risk scores	Racial discrimination	No subgroup confusion matrix or uncertainty analysis
Employment	Amazon résumé screening (2018)	Female-associated terms were downgraded	Gender bias	No feature attribution audit or shortlist parity check
Healthcare	Risk prediction model (2019)	Black patients underreferred despite equal risk	Proxy-label bias	No label validity check or subgroup error analysis
Recruitment	Automated screening lawsuit (2022)	Older and Black candidates excluded	Age and race discrimination	No slice-level evaluation or intersectional coverage
Lending	Credit scoring disparities (2021)	Minority borrowers received worse terms	Disparate impact	No distributional stress test or sensitive attribute guardrails

Table 1.2: Disparities in COMPAS Risk Labeling and Rearrest Rates by Racial Group (ProPublica Analysis)

Group	**Labeled High Risk (%)**	**Actual Rearrested (%)**
Black	45	20
White	23	13

Table 1.3: *

Source: Adapted from ProPublica analysis (2).

pdflscape adjustbox float array

Table 1.4: Illustrative Responses Linking Incidents to Governance Requirements
max width=

Incident/Domain	**Litigation/Regulatory Action**	**New or Reinforced Requirement**
HR screening bias	Class-action filings; agency inquiries	Audit logs for automated hiring, fairness documentation, pre-deployment bias testing
Credit scoring discrimination	Consumer-protection and DOJ enforcement	Fair-lending model validation, explainability, disparate-impact monitoring
Healthcare risk discrimination	Health-system compliance initiatives	Equity audits, algorithm review boards, proxy-label testing

Practical takeaway: Fairness testing must become a natural, world-class requirement in addition to accuracy, reliability, and security. Set slice coverage, employ uncertainty-aware thresholds, offer mutation scores with confidence intervals, and maintain artifacts showing that fairness-altering mutants would have been detected before deployment. It is through overloading our test environments with fairness-aware protocols and dynamic adequacy that we can safeguard both our algorithms and our collective ethics against the next wave of systemic, but preventable, failures.

1.6 Importance of Solving This Problem

There are a lot of good arguments for the machine learning to be fairminded in the first place, especially when it comes to such sensitive areas as credit scoring, policing and even healthcare. An AI which is fair by design helps to negate any harm, build trust and moral acceptability, and make sure that all demographic groups get their fair share of resources and opportunities. For this reason, dealing with fairness is not only a technical challenge but also a requirement for the responsible and sustainable deployment of AI.

Besides that, the technical side of the story, fairness testing has the direct advantage of revealing flaws in the system that may otherwise be not obvious due to the overall performance metrics hiding them. It promotes systematic error analysis across different subgroups and features contexts instead of depending on a single global accuracy score. In this thesis, fairness becomes a *test adequacy requirement* rather than merely a diagnostic afterthought: we inquire if the test suite is sufficiently robust to uncover fairness-related bugs prior to deployment.

To put our approach into practice, we simulate small, realistic data changes under distributional constraints and check if there is any change in the fairness metrics

detection by a frozen test suite. A mutant is called *killed* when there is an absolute change in the fairness metric beyond the threshold derived from the base model's null variability (mean + 1.5 std; see Section **??**). The mutation score obtained indicates how well the test suite is open to fairness defects through sensitive slices and possible shifts, and is the foundation for an adequacy-style certificate that states whether the fairness is still within the permissible limits.

These demands emphasize that fairness should be regarded as the primary testing focus in the development of predictive systems, not just an optional post hoc check. Embedding fairness adequacy into the testing process is our way of fostering not only technical robustness but also the broader social expectation that automated decision systems will be free from discrimination.

1.7 Related Work

The main issue of fairness in machine learning has an interdisciplinary character involving computer science, law, and policy. The collaboration of a multidisciplinary community consisting of researchers from different fields, the industry, and law makers has been a source of both theory and tools. However, the quantification of *fairness test adequacy* still remains as an unexplored area. The following sections will present the main topics of our interest.

Foundational fairness definitions and trade-offs. Feldman et al. (13) start with detecting disparate impact as their first step and then move on to their pre-processing repair to certify and reduce discrimination between the groups. Hardt et al. (17) propose the concept of Equality of Opportunity (EO) defined as equal true-positive rates across all groups thus crystallising the fairness paradigm of EO/Equalized Odds that is now widely adopted in the fairness literature. Chouldechova (8) has shown

that calibration and equalised error rates can generally not be achieved together if the base rates are different, thus motivating transparent metric choice. Kusner et al. (**?**) provide an approach to counterfactual fairness using causal models: the decisions should not change when the protected characteristics are altered in a random manner.

Fairness testing, auditing, and "fairness bugs." *Themis* (14) introduces the idea of *fairness bugs* and presents testing workflows that help in revealing discriminatory practices. The *FairTest* (Tramèr et al.) (38) method identifies statistically significant *unwarranted associations* and provides support for debugging to rule out confounders. Toolkits like IBM's *AI Fairness 360* (5), Microsoft's *Fairlearn* (39), and *Aequitas* (21) establish standard metrics, mitigation strategies, and practitioner reports.

Model Cards (27) suggest reporting not only the performance of a model for a certain subgroup but also the intended use and limitations.

Datasheets for Datasets (15) focus on documenting the collection, coverage, and even biases, which in turn makes it easier to estimate slice coverage.

Arbie et al. (19) Are of the opinion that the Adult dataset is so overused that it should be put to rest and new benchmarks should be created. They point out that preprocessing can cause fairness conclusions to be distorted without the researcher realizing it.

Shift diagnostics and null variability. The study of label-shift detection and correction, together with empirical investigations of dataset-shift detectors (35), supports the argument for separating genuine fairness alterations from distributional artifacts. This gives us guidelines for our distribution and bootstrap-based null estimation.

Mutation and metamorphic testing for ML robustness. Differential and metamorphic testing for deep neural networks (DNNs) apply structured input changes to bring out incorrect behaviors without the necessity of completely defined oracles. DeepTest (4) uses metamorphic transformations to uncover faults that are not immediately acknowledged in the DNNs for autonomous-driving. DeepMutation (24) organizes mutation testing for DNNs with source- and weight-level operators and looks into their ability to reveal faults.

Empirical failures motivating fairness evaluation. ProPublica's COMPAS scrutiny (2) gives the account of Black defendants suffering from the highest false-positive rates, thereby setting the stage for the adoption of EO/EOdds and subgroup confusion matrices. Obermeyer et al. (30) demonstrate that one of the most widely used clinical risk algorithms misinterpreted *cost* as a proxy for *need*, thus under-referring Black patients and exposing the shortcomings of proxy-labels.

Testing and adequacy in ML. Myers and colleagues (29) suggest that adequacy in testing should be viewed as the coverage of code and paths that are likely to reveal faults in classical software testing. Sharma and Wehrheim (37) introduce a perspective on ML testing as property-based: the non-functional properties (fairness, robustness, etc.) must be specified and tested just like functional requirements. Grote (16) pushes the boundaries of the notion of adequacy with a sociotechnical view, indicating that evaluation should not only be based on laboratory metrics but also on real-world impacts.

How this work differs. Audits and toolkits are mainly used to address the problem of *"is this model biased?"*, while robustness research applies mutation to detect accuracy bugs. Our attention is drawn to a *quantified notion of fairness test adequacy*:

do tests over slices, contexts, and realistic shifts consistently *kill fairness-affecting mutants*? We, in a more concrete way, propose (i) a mutant-validity gate with distribution checks and a training-set fairness shift, (ii) a bootstrap-based kill threshold (mean + 1.5 std) to distinguish the signal from the noise, and (iii) different test sets (balanced, skewed, group-dropped) for coverage and detection power analysis.

1.8 Why Resolving This Is Important

Fairness is no longer optional. In high–stakes deployments, failing to test for subgroup harms invites lawsuits, product rollbacks, and loss of public trust. A defensible program must provide *auditable evidence* that foreseeable biases would have been caught before deployment.

A recent, concrete case: AI hiring under legal scrutiny

In 2024–2025, *Mobley v. Workday* placed automated résumé screening under national scrutiny. The complaint alleges that an AI-driven filtering system disproportionately excluded older and Black applicants, and a federal court allowed core claims to proceed, signaling potential liability for vendors as well as employers. The case is shaping how providers and deployers must validate automated screening tools, with courts and regulators expecting documented, pre-deployment fairness evaluation (28).

What went wrong in evaluation terms. Filings and expert commentary point to three gaps that rigorous fairness testing would have addressed:

(i) insufficient *slice coverage* across protected and intersectional groups during evaluation.

(ii) weak *shift sensitivity* to small cohorts and feature sparsity patterns by age or race.

(iii) missing *audit artifacts* that tie fairness requirements to concrete tests and thresholds.

Had these been in place, the vendor and customers could have shown evidence that group-level disparities would be detected pre-deployment.

Why this matters beyond one lawsuit

Policy is converging on documented bias controls. The EU AI Act imposes lifecycle obligations for high-risk systems, including risk management, data governance, logging, and post-market monitoring (12). In U.S. credit, the Consumer Financial Protection Bureau requires specific adverse-action reasons even when AI is used (10; 9). Organizations that cannot produce test artifacts face enforcement risk and slower adoption.

How our approach closes the gaps

This thesis treats fairness as *test adequacy*, not just metric reporting. We:

1. Build explicit **slice coverage** (age, race, gender, intersections) into the test suite.

2. Generate **realistic mutations** (cohort reweighting, proxy-feature stress, sparse résumé sections) to simulate shifts that induce fairness faults.

3. Enforce a **mutant-validity gate** (distribution similarity plus training-set fairness shift) to avoid meaningless mutants.

4. Apply a **data-driven detection threshold** using the base model's bootstrap null (mean + 1.5 std; see Section **??**) and declare a mutant *killed* when the fairness change exceeds that threshold.

5. Summarize detection power with a **Mutation Score** and confidence intervals, producing *audit-ready artifacts* (test suites, operators, thresholds, logs).

Counterfactual: preventing a Workday-style failure. In a résumé-screening pipeline, our mutations would (a) up- and down-weight applicant segments common among older candidates, (b) jitter or mask proxy features for age or race (for example, graduation-year ranges, tenure patterns), and (c) simulate employer filters. If the suite fails to kill a sufficient fraction of these fairness-affecting mutants (Mutation Score below target), we fail the build and emit a remediation report that lists missing slices, fragile features, and data-collection needs.

Industry and national relevance, with evidence

Hiring and employment. Active litigation is redefining responsibilities for vendors and deployers. Maintaining test artifacts mitigates legal and reputational risk (28).
Healthcare. A widely used clinical risk model under-referred Black patients because cost was used as a proxy for need; our operators explicitly perturb proxy labels and cohort mix to reveal such failures before rollout (30).
Finance. When AI supports credit decisions, creditors must provide specific adverse-action reasons; our artifacts tie slice-level results to compliant explanations (10; 9).
Regulation. The EU AI Act codifies measurable, documented bias controls for high-risk domains; our approach produces the needed evidence trail (12).

Bottom line

Fairness failures are now triggering legal and regulatory action, not only academic debate. A mutation-based adequacy protocol turns fairness from an aspiration into an *operational gate* with measurable detection power and auditable proof, so unfair

outcomes are eliminated before they scale.

1.9 Limitations Observed in This Study

Our empirical analysis surfaced several practical constraints and threats to validity. We list the most consequential ones here.

L1. One-class / extreme class imbalance (core limitation). *What we observed:* In some datasets and slices, the positive class was extremely rare (near one-class behavior), and in a few intersectional subgroups it was absent.
Why it matters: Group fairness metrics (e.g., EO/EOdds) become high-variance or undefined; ROC/AUC can be misleading under severe imbalance.
What we did / will do: Used stratified bootstrap with minimum-support filters, Wilson intervals for rates, and conservative CI reporting; for empty slices we reported "insufficient support" and recommended targeted data collection. Future work will add reweighting/SMOTE variants gated by distribution tests.

L2. Sparse intersectional slices. *What we observed:* Many age×race×gender intersections had very small n.
Why it matters: Low power to detect disparities; instability in mutation deltas.
What we did / will do: Applied a slice-coverage manifest with minimum count thresholds; merged adjacent levels where policy-acceptable; flagged under-powered slices in the remediation report.

L3. Limited sensitive-attribute availability. *What we observed:* Some attributes were missing or available only via coarse proxies.
Why it matters: Risk of proxy bias and under-detection of harms.

What we did / will do: Ran proxy-feature sensitivity checks; documented attribute gaps; recommended safe data governance paths for improved coverage.

L4. Narrow metric panel. *What we observed:* Primary analysis focused on Equality of Opportunity / Equalized Odds.
Why it matters: Models can satisfy these yet fail under calibration or individual/-counterfactual fairness.
What we did / will do: Added calibration by group as a robustness check; plan to include one individual/counterfactual proxy in follow-ups.

L5. Mutation operator coverage. *What we observed:* Operators targeted training-data perturbations for tabular ML; no model-architecture mutations and no modality-specific transforms (e.g., text/vision).
Why it matters: Adequacy estimates depend on mutation realism; coverage gaps can understate risk.
What we did / will do: Instituted a mutation-design audit (face validity + domain sign-off); plan modality-specific operators in a small cross-domain pilot.

L6. Mutant-validity gate tuning. *What we observed:* Distribution-similarity tests and training-fairness shift gate occasionally rejected subtle but plausible shifts; conversely, some accepted mutants were borderline.
Why it matters: Over-rejecting reduces realism; over-accepting inflates detections.
What we did / will do: Reported acceptance/rejection rates per operator; added a manual review bucket; will calibrate gates against historical shift examples.

L7. Detection threshold sensitivity. *What we observed:* Initial formula was corrected to a bootstrap-null threshold (mean + 1.5 std); results vary with the mul-

tiplier and B.

Why it matters: Conclusions can shift under different, reasonable settings.

What we did / will do: Conducted sensitivity sweeps over multipliers $[1.0, 2.0]$ and B resamples; will include multiple-testing adjustments in reporting when monitoring many metrics×slices.

L8. Compute budget for mutants. *What we observed:* Full retraining across many mutants is costly for CI cadence.

Why it matters: Limits breadth of operators and frequency of checks.

What we did / will do: Adopted a two-stage screen (subsample/surrogate to prune; full retrain for survivors) and cached artifacts for incremental runs.

L9. Offline evaluation only. *What we observed:* All evidence is from frozen test suites; no shadow-mode or online canary yet.

Why it matters: Real deployments include drift, feedback loops, and strategic behavior.

What we did / will do: Specified a shadow-mode plan (non-impacting traffic + fairness alerts) and measurement of offline→online concordance.

L10. Baseline comparisons. *What we observed:* Limited head-to-head comparison with Fairlearn/AIF360/FairTest or counterfactual methods.

Why it matters: Reviewers expect relative positioning and complementary strengths.

What we did / will do: Prepare a comparative study correlating our Mutation Score with baseline detections and documenting cases each method uniquely finds.

L11. Governance and audit trail mapping. *What we observed:* Artifacts not yet mapped one-to-one to domain obligations (e.g., adverse-action reasons, post-market

monitoring).

Why it matters: Harder for auditors to sign off; slows deployment.

What we did / will do: Add a requirements traceability matrix and an audit pack (slice manifest, operator configs, thresholds, per-slice deltas, remediation notes).

1.10 Contributions

C1. Adequacy certificate (maps to RQ1): an ε-tolerant, mutation-based certificate returning **ALIVE**/**KILLED** decisions from data-driven control limits on fairness metrics.

C2. Operators & guardrails (RQ1): fairness-aware mutation operators with checks for distributional and slice support; we report both MKR and MKR_{adj}.

C3. Cross-model/dataset study (RQ2): testing on Logistic Regression, kNN, and Decision Tree models with Adult and COMPAS datasets, repeating test executions to account for uncertainty.

C4. Baselines (RQ3): competing directly against Aequitas/Fairlearn and a naïve model; *FairPipes operators are used solely as comparison baselines and are not part of our method.*

C5. Artifacts & presentation: reproducible suites/configurations and an audit pack. In our results, we show only the **top-5** sensitive attributes in the main text; full panels are provided in the appendix. All figures adhere to a caption/axis/legend checklist.

Summary. We make intersectional fairness practical and testable, providing statistical scaffolding and engineering artifacts that go beyond "is there bias?" to "is the test suite adequate under realistic shifts?"—with **audit-ready evidence for**

industry and regulators, as demonstrated on public datasets and real-world case applications.

Chapter 2

Background

2.1 Introduction

Machine learning (ML) has gone through a whole journey from being an academic interest to a dominant technology that influences decisions in finance, healthcare, hiring, and even criminal justice. In these areas, algorithmic predictions are not merely recommendations; they are part of the decision-making process and thus, affect the lives of people directly. The chapter that follows places our research in the context of ML research as a whole, stressing the importance of fairness adequacy in supervised classification.

2.2 Machine Learning in High-Stakes Decisions

Initially, ML technology was applied to areas associated with low risk like spam filtering, handwriting recognition, and recommendation systems. However, nowadays ML models are indispensable in critical areas:

- **Finance:** The algorithms that score credit determine not only the approval of a loan but also its interest rate.
- **Healthcare:** The models that predict the future help in diagnosis, deciding who gets treated first, and even treatment prioritization.

- **Hiring:** Automatic screening of resumes affects the candidates' chance of getting hired.
- **Criminal Justice:** Tools for assessing risk make recommendations regarding parole and, in a way, sentencing.

The use of such technology raises the moral issues of fairness and accountability to an extreme degree (3; 31). One of the common methods for conducting a fairness audit is to use group metrics like Statistical Parity Difference (SPD), Equal Opportunity Difference (EOD), and Equalized Odds Metric (EOM) which are calculated on a single test set at a predetermined threshold (17). Nevertheless, the problem with such static evaluations is that they are very sensitive, as the data distributions shift, the thresholds drift, and the features evolve, hence, the fairness could deteriorate even if the model remains the same (34; 26).

The conditions under which a model works in the real world are seldom fixed. The input distributions can change due to the alteration of demographic factors or the implementation of new policies; besides, the thresholds are frequently adjusted to align with the business goals; also, the feature definitions change as the data pipelines upstream get modified. It has been proven in various studies that predictive performance and fairness metrics fall back under even slight distribution shifts (34; 26). What is more, looking only at rough demographic divisions may hide discrimination in terms of intersections or small groups of people (22).

These insights give rise to the main idea of this dissertation, *fairness adequacy*. Our focus is not whether the model will ever be considered fair after a certain evaluation has been conducted, but rather if, in a realistic, domain-plausible - a portfolio of perturbations - the fairness issue is still within the acceptance range. The Fairness Adequacy Test (FAT) is basically a mutation-testing way of thinking: perturbing the

training data and model inputs in a predetermined way, and then checking if a frozen fairness test suite consistently identifies fairness-relevant changes before the model is deployed.

2.3 Formal Machine-Learning Pipeline

An ML algorithm approximates a function

$$f : \mathcal{X} \to \mathcal{Y}$$

from experience $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$

The $\mathcal{D}$ bias incorporates past unfairness; the f that is learned may then cause protected groups to be systematically disadvantaged. This thesis deals with the binary classification in supervised learning throughout its duration, the scenario where fairness metrics are developed the most and are under legal scrutiny.

2.4 Base Learners in Our Experimental Campaign

We proceed to present an extensive review of the three algorithms that were employed in the mutation campaign: k-NN, Decision Tree, and Logistic Regression. For each, we make available the following: the inductive bias, operational pseudocode, computational footprint, fairness sensitivities, concrete failure example, hyper-parameter grid, and expected mutant-kill signature

2.4.1 Logistic Regression (LR)

Logistic Regression is one of the most widely used linear classifiers in statistics and machine learning. Originally introduced by (**?**) as a model for binary outcomes, it has become a cornerstone in applied domains such as credit scoring, medical diagnosis,

and risk assessment due to its interpretability and efficiency. Unlike linear regression, which predicts continuous values, logistic regression models the probability of a binary outcome using the logistic (sigmoid) function.

$$\hat{p}(y = 1|x) = \sigma(w^\top x + b), \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

Here, w denotes the vector of feature weights, b the bias term, and $\sigma(\cdot)$ the sigmoid function that maps real values to the interval $(0, 1)$. The decision boundary is defined by the hyperplane $w^\top x+b = 0$, which separates the feature space into regions classified as positive or negative.

Training. Parameters (w, b) are estimated by maximizing the likelihood of the observed data, equivalently minimizing the negative log-likelihood. To prevent overfitting and encourage sparsity or smoothness, regularization terms are added:

$$\mathcal{L}(w) = -\sum_{i=1}^{n} \Big[y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i) \Big] + \lambda \|w\|_p,$$

where $\|w\|_p$ denotes either the ℓ_1 (Lasso) or ℓ_2 (Ridge) penalty. Optimization is typically performed using Newton's method, coordinate descent, or gradient descent.

Algorithm 1: Logistic Regression (Newton/CD)

Input : Data $\mathcal{D}$, regularisation λ, convergence tol ϵ

Output: Weights (w, b)

repeat

  $w \leftarrow w - \eta \nabla(\mathcal{L}_{\text{log-likelihood}} + \lambda \|w\|)$

until $\|\nabla \mathcal{L}\| < \epsilon$;

Complexity. Training requires $O(nd^2)$ operations with Newton's method or $O(ndT)$ with gradient descent (where T is the number of iterations). Prediction is efficient at $O(d)$ per query, with memory footprint $O(d)$.

Interpretability. A key advantage of logistic regression is its transparency: each coefficient w_j can be interpreted as the log-odds change in the outcome per unit increase in feature x_j, holding other features constant. This interpretability makes LR particularly attractive in regulated domains where explanations are legally mandated.

Fairness sensitivity. Because logistic regression imposes a global linear boundary, any non-zero weight on a protected attribute or its proxy propagates bias across the entire feature space. For example, if x_2 is correlated with gender, a positive weight $w_2 > 0$ tilts the separating hyperplane, lowering predicted scores for female applicants everywhere. This results in a Statistical Parity Difference (SPD) violation that cannot be corrected by threshold tuning alone.

Concrete failure. In Figure **??**, the orange line represents the LR decision boundary. If the model learns a spurious correlation between x_2 and gender, all female instances are systematically disadvantaged, even those far from the majority cluster. This illustrates how global linearity can amplify bias.

Hyper-parameters. Key hyper-parameters include the penalty type $\{\ell_1, \ell_2\}$, inverse regularization strength $C \in \{0.01, 0.1, 1, 10\}$, class-weight balancing, and fairness-regularised objectives (**?**). These settings control the trade-off between accuracy, sparsity, and fairness.

Mutant kill signature. Logistic regression is most sensitive to covariate-shift mutants that rotate the linear separator away from minority directions. Because the model enforces a single global boundary, even small distributional changes can disproportionately affect protected groups.

2.4.2 Decision Tree CART

Decision Trees are surprisingly still one of the most interpretable and commonly employed supervised learning techniques. The Classification and Regression Tree CART concept, presented by (6), formalizes the cutting up of the feature space along the axes (6). The method takes a feature and a threshold that produces the greatest decrease in impurity, which is usually evaluated by Gini index, entropy or variance reduction, at every node. This greedy splitting process goes on until a stopping rule is applied, for instance, the maximum depth or minimum samples per leaf.

Training. The CART process starts at the root node with the entire dataset. It considers the impurity for each potential split (j, t)

$$\Delta I = I(\mathcal{D}_{\text{node}}) - \frac{|\mathcal{D}_{\text{left}}|}{|\mathcal{D}_{\text{node}}|} I(\mathcal{D}_{\text{left}}) - \frac{|\mathcal{D}_{\text{right}}|}{|\mathcal{D}_{\text{node}}|} I(\mathcal{D}_{\text{right}}).$$

The split with the highest ΔI is selected, and the same procedure is applied recursively to the left and right child nodes. The majority class (in case of classification) or the mean value (in case of regression) is assigned to the leaves.

Complexity. It takes $O(nd \log n)$ time to train the model with n samples and d features, assuming sorted continuous features. Prediction is fast and requires just $O(\text{depth})$ comparisons for every query. The memory consumption keeps growing based on the number of nodes.

Interpretability. Decision Trees are appreciated for their clarity: the model trained can be represented as a tree and the predictions can be understood by following the path from the root to the leaf. This quality makes them desirable in the areas where interpretability is of utmost importance, for example, in medicine and legal affairs.

Fairness sensitivity. On the one hand, interpretability is an advantage of decision trees; on the other hand, they are vulnerable to fairness problems. Greedy splitting optimizes for accuracy and does not consider fairness, allowing proxy discrimination to be built into the structure. The minority can be pushed into very small leaves and then the class frequencies can be quite different from that of the majority. If pruning cuts those leaves, then the predictions may follow the majority class and this will worsen the inequality.

Concrete failure. A single cut such as $x_1 \leq 3$, for instance, as shown in Figure **??**, can move all red (minority) points to one child node. If that node is subject to pruning by minimum leaf constraints, its predictions will be the majority ones (blue), thus pushing the false positive rates for the minority group up to 100%. This is a clear case of how unfairness can stem from the world of tree structure decisions.

Hyper-parameters. The major hyper-parameters consist of the maximum depth, the minimum number of samples per leaf, and the criterion for splitting (Gini or entropy). Balancing the class weights

2.4.3 k-Nearest Neighbours (k-NN)

k-Nearest Neighbours (k-NN) is one of the non-parametric methods and is still the one most commonly used because of its simplicity and induction bias based on locality (**?**). In contrast to parametric models, k-NN does not presume a certain functional

form; rather, it classifies observations in a new location based on the nearby training points whose labels are considered in feature space.

Training. k-NN is a "lazy learner": it does not require an explicit training stage except for keeping the dataset. When a query point x_* is given, the distances to all training points are calculated and the k nearest neighbours are selected. In case of classification, the label $\hat{y}_*$ that is predicted for the query point is obtained through majority voting among the neighbours; while in regression, the prediction is simply the mean of the target values of the neighbours considered.

Distance metrics. The selection of the distance function is a very important factor. The commonly used metrics are Euclidean distance, Manhattan distance, and cosine similarity. In the case of weighted metrics, more powerful effects are given to closer neighbours. Often, feature scaling and normalization are necessary to avoid the situation where the features with larger ranges dominate the calculation of distances.

Complexity. The training complexity is $O(1)$ (only store the dataset). Query complexity for brute-force search is $O(nd)$ where n is the number of samples and d is the dimensionality. In low dimensions, query time can be reduced to $O(\log n)$ using KD-trees or Ball Trees; however, the performance declines in high-dimensional spaces because of the curse of dimensionality.

Interpretability. k-NN is a very simple concept: the predictions are derived from the local neighborhoods. As a result, it is simple to justify the results using "similar past cases." Nonetheless, this interpretability fades away in high-dimensional feature spaces where metrics of distance lose their significance.

Fairness sensitivity. K-NN, which works on local majority voting, has the potential to enhance the bias that has been clustered. In a situation where the instances of a minority group are entirely surrounded by majority group points, their labels could be consistently outvoted, causing systematic disparities. This sensitivity is extremely pronounced in areas of feature space wherein the representation of protected groups is scant.

Concrete failure. Refer to Figure 2.1, where the red crescent shape corresponds to a minority protected group. When $k = 5$, the red points located at $(2, 2)$ are surrounded by five blue points. The true positive rate of the red group will be reduced to zero, while the blue group continues to perform at a high level, resulting in an Equal Opportunity Difference (EOD) infringement. It is worth noting that such failures occur even after a small covariate shift, which is an indicator of how fragile fairness is in k-NN.

Hyper-parameters. The main hyper-parameters are the number of neighbours $k \in \{3, 5, 7, 11, 21\}$, the method of giving weights (uniform vs. distance-weighted), the type of distance metric (Euclidean, Manhattan, learned metric), and the ways of preprocessing (normalization, dimensionality reduction). Fairness-aware variants apply either re-weighting of neighbours or adversarial debiasing (**?**).

Mutant kill signature. k-NN is indeed the most impacted by feature-noise mutants, since distance calculations are directly affected by the perturbations. Neighborhood composition can be changed by just small modifications in feature values, resulting in prediction and fairness outcome flipping.

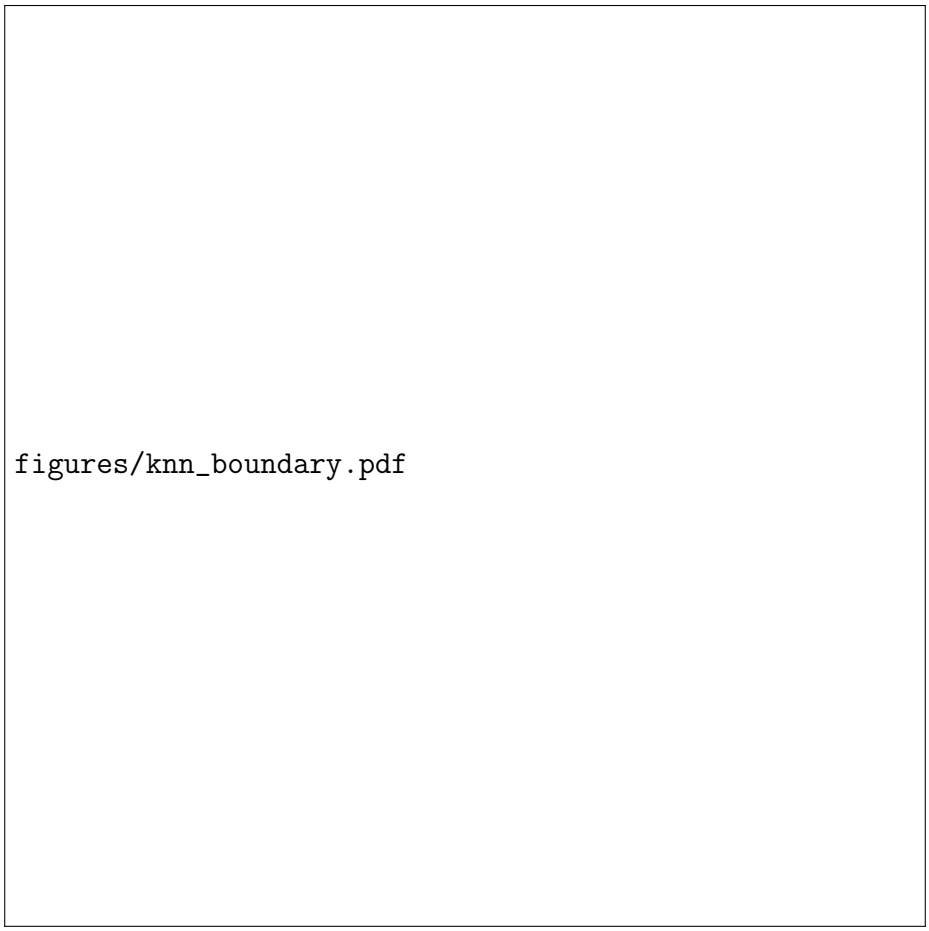

Figure 2.1: Decision boundary induced by k-NN. Local neighborhoods determine classification, which can magnify clustered bias.

2.4.4 k-Nearest Neighbours (k-NN)

Algorithm 2: k-NN Prediction

Input : Training set $\mathcal{D}$, query x_*, integer k, distance $\mathrm{dist}(\cdot,\cdot)$

Output: Predicted label $\hat{y}_*$

$N_k \leftarrow \arg\min_{\mathcal{S}\subseteq\mathcal{D},|\mathcal{S}|=k} \sum_{(x,y)\in\mathcal{S}} \mathrm{dist}(x, x_*)$;

return `majority-vote`$\{y : (x,y) \in N_k\}$;

The capacity ladder is LR *(low)* → Tree *(medium)* → k-NN *(high)*; hence mutation adequacy becomes more critical as flexibility grows. All hyper-parameter grids above

Table 2.1: Big-O summary (n samples, d features).

	Training	Query	Memory
k-NN	$O(1)$	$O(nd)$	$O(nd)$
Decision Tree	$O(nd \log n)$	$O(\log n)$	$O(\#\text{nodes})$
Logistic Regression	$O(nd^2)$	$O(d)$	$O(d)$

are exactly those used in **??**; mutant-design that targets each model's kill-signature is detailed in **??**.

2.5 Mutation Testing for Fairness

2.5.1 Core Concepts

Mutation testing injects small, controlled changes (mutants) into data or pipeline and checks whether the fairness oracle flags the change—mimicking software mutation scores.

2.5.2 Kill / Alive Decision Rule

For fairness metric f (EOD, EOM, SPD) we compute

$$\Delta f_m = f_m - f_{\text{base}}.$$

Using $B = 1000$ bootstrap resamples of the test set we estimate $\mu_{\text{null}}, \sigma_{\text{null}}$ under no change. Mutant m is KILLED if

$$|\Delta f_m| \;\geq\; \mu_{\text{null}} + 1.5\,\sigma_{\text{null}} \;=\; \theta_f,$$

otherwise ALIVE (Figure 3.1). When all monitored metrics slices remain below their θ the mutant survives.

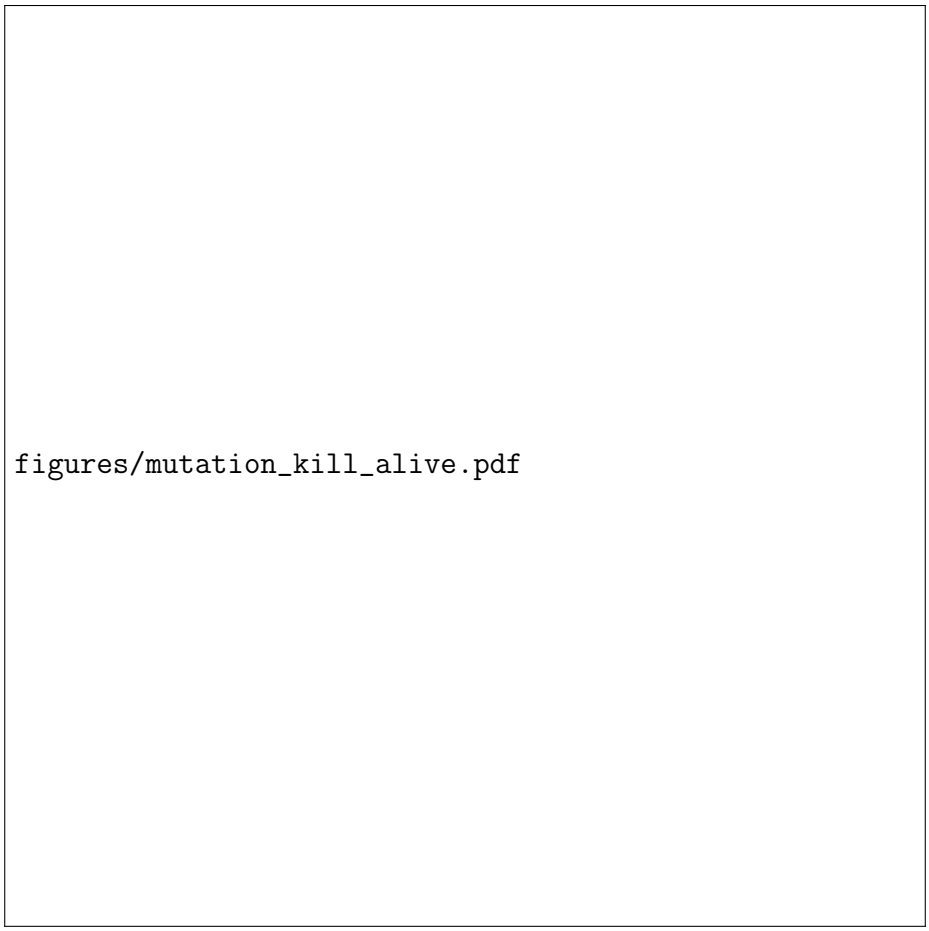

Figure 2.2: Fairness mutation lifecycle. **Killed**: $|\Delta\text{EOD}| \geq \theta$. **Alive**: below threshold (possibly equivalent).

2.6 Mutation Testing for Fairness

Mutation testing originated in software engineering as a method to evaluate test adequacy (**?**). The central idea was to introduce small, controlled changes (mutants) into a program's source code and then check whether the existing test suite could detect the change. If the test suite failed to distinguish the mutant from the original program, the mutant was said to be *alive*; if the test suite detected the difference, the mutant was *killed*. This framework provided a rigorous way to measure the strength of a test suite beyond simple coverage metrics.

Analogy to Machine Learning Fairness. In the context of machine learning, we adapt mutation testing to evaluate the adequacy of fairness audits. Here, the "program" is the trained ML model and its decision pipeline, while the "test suite" is the collection of fairness metrics applied to the model outputs. Mutations correspond to controlled perturbations of the data, features, labels, or thresholds that simulate realistic distributional shifts. The fairness audit is considered adequate if it reliably detects fairness-relevant changes introduced by these mutants.

Motivation. Traditional fairness evaluation computes metrics such as Statistical Parity Difference (SPD), Equal Opportunity Difference (EOD), and Equalized Odds Metric (EOM) on a single static test set. However, real-world deployments are dynamic: input distributions drift, thresholds are tuned, and features evolve. A fairness audit that appears satisfactory under one snapshot may fail under modest perturbations. Mutation testing provides a principled way to stress-test fairness audits before deployment, ensuring that fairness violations are not silently overlooked.

Mutation Operators. We define several classes of mutants relevant to fairness:

- **Feature-noise mutants:** Small perturbations to input features, simulating measurement error or preprocessing drift.
- **Label-flip mutants:** Controlled mislabeling of training data, representing annotation bias or adversarial corruption.
- **Covariate-shift mutants:** Reweighting or resampling of subpopulations, mimicking demographic change.
- **Threshold mutants:** Adjustments to decision thresholds, reflecting operational tuning.

Each operator is designed to mimic realistic conditions under which fairness may degrade.

Kill/Alive Rule. For fairness metric f (e.g., EOD, SPD, EOM), we compute the difference between the mutant and baseline:

$$\Delta f_m = f_m - f_{\text{base}}.$$

Using $B = 1000$ bootstrap resamples of the test set, we estimate the null distribution $(\mu_{\text{null}}, \sigma_{\text{null}})$ under no change. A mutant m is **KILLED** if

$$|\Delta f_m| \geq \mu_{\text{null}} + 1.5\sigma_{\text{null}} = \theta_f,$$

otherwise it is **ALIVE**. Killed mutants indicate that the fairness audit successfully detected the perturbation; alive mutants suggest inadequacy.

Worked Example. Suppose the blue group has TPR = 0.80 and the red group has TPR = 0.62. The Equal Opportunity Difference (EOD) is 0.18. If the bootstrap threshold is $\theta_{\text{EOD}} = 0.10$, then the fairness audit detects a violation and the mutant is killed. If the difference were only 0.05, the mutant would survive, indicating that the audit may miss subtle but consequential disparities.

Lifecycle. Figure 3.1 illustrates the fairness mutation lifecycle. Mutants are generated by perturbing the data or pipeline, evaluated by fairness metrics, and classified as killed or alive depending on whether thresholds are exceeded. This lifecycle mirrors the software engineering paradigm, but is adapted to the fairness domain.

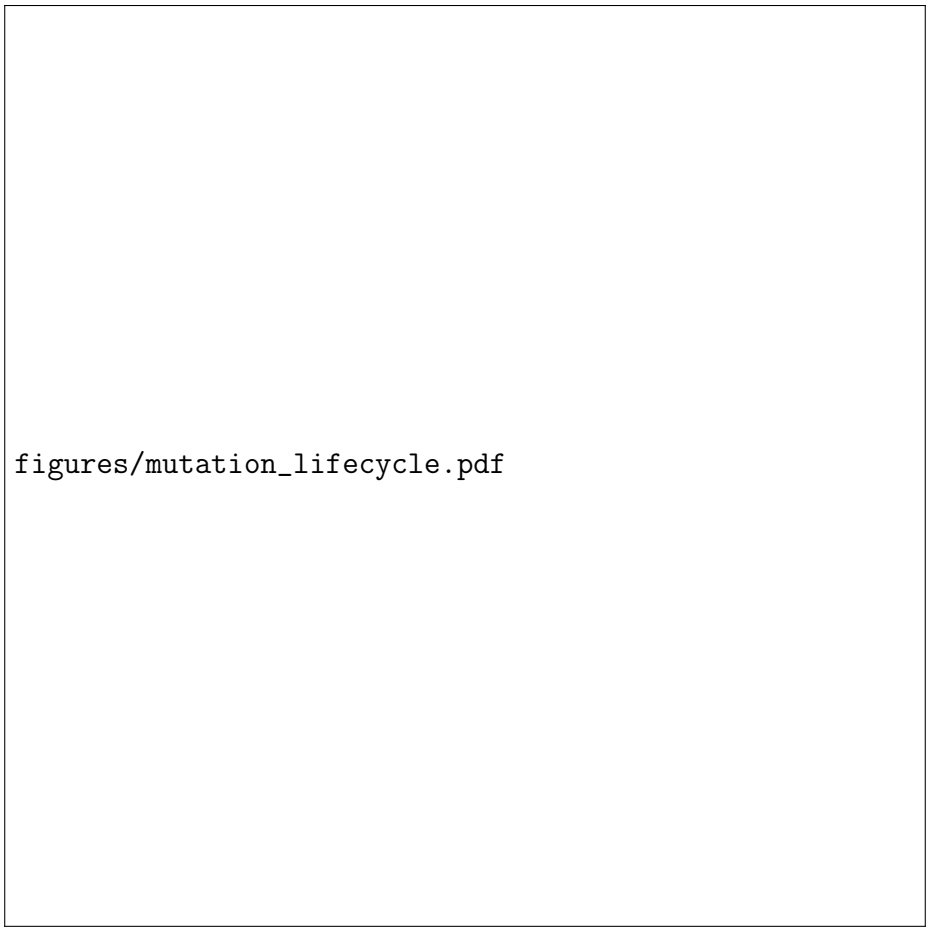

Figure 2.3: Fairness mutation lifecycle. Mutants are killed if fairness metrics exceed bootstrap thresholds.

Implications. By systematically applying mutation testing to fairness audits, we gain a measure of *fairness adequacy*. This goes beyond static evaluation, probing whether fairness metrics are robust to realistic perturbations. A fairness audit with high mutation kill rates can be considered adequate; one with many surviving mutants is inadequate, even if static metrics appear acceptable. Thus, mutation testing provides a principled foundation for fairness stress-testing prior to deployment.

2.7 Summary of Background Needs

We have:

(i) positioned ML fairness in high-stakes deployment;

(ii) provided extensive, pseudocode-backed descriptions of k-NN, Tree, LR with Big-O, failure examples, and mutation signatures;

(iii) illustrated kill/alive rule via figure;

(iv) listed formal fairness metrics with worked numeric example.

The next chapter translates these ingredients into the Fairness Adequacy Testing framework and the mutation operators that realise the kill rule.

Chapter 3

Method: Fairness-Oriented Mutation Testing

3.1 Introduction

This chapter describes the structured, mutation-driven methodology for assessing the fairness of machine learning (ML) models. Building on the Fairness Adequacy Test (FAT) introduced in previous chapters, we translate classical software mutation testing into the fairness realm: instead of injecting faults into source code, we inject small, *domain-plausible* perturbations (mutants) into the *data* or *pipeline*. A fixed fairness oracle (based on group fairness metrics) is then run on the *same test set* for the base and mutant models. If *any* fairness metric exceeds a *data-driven control limit*, the mutant is labelled **KILLED**; otherwise it remains **ALIVE**.

Formally, let M_{eff} denote the number of non-equivalent mutants (mutants that are capable of changing fairness on the test set). The resulting adequacy score is the *Mutation Kill Rate* (MKR),

$$\text{MKR} = \frac{\#\,\text{KILLED mutants}}{\#\,\text{non-equivalent mutants}} = \frac{\#\,\text{KILLED}}{M_{\text{eff}}},$$

which we interpret as a fairness-oriented test-adequacy score. Higher MKR values indicate that the fairness test suite is more sensitive to fairness-relevant changes and therefore more adequate.

Throughout, we evaluate MKR for the three base learners introduced in Chapter 2 (k-NN, Decision Tree, Logistic Regression) and for the fairness metrics defined there (SPD, EOD, EOM).

3.2 Methodology Workflow

The methodology is an iterative process that combines fairness metrics and mutation testing to attain fairness adequacy. The overall workflow is shown in Figure 3.2.

The main steps are:

1. **Identify dataset.** Select a dataset that contains relevant sensitive attributes such as gender, ethnicity, age, and socioeconomic status. The dataset is the basis for generating test cases and assessing fairness violations.

2. **Generate mutants for training data.** Create mutants via perturbations on the dataset, including feature changes. The mutations simulate changes in sensitive attributes so that their impact on model fairness can be examined. The mutation operators used are detailed in Section 3.5.

3. **Train and evaluate the ML model.** Train the machine learning model with the original and mutated datasets. The trained model is then tested with fairness measures to determine potential biases introduced by different sensitive-attribute variations.

4. **Apply fairness group metrics.** Apply group-fairness measures such as demographic parity, equalized odds, and equal opportunity to analyse how the model's predictions change across subgroups. These measures help to quantify bias in the model's decisions (Section 3.4).

5. **Determine mutant status (killed or alive).** A mutant is labelled as **killed** if it has a significant impact on fairness measures, i.e. it reveals a fairness violation. If the mutant has no effect on fairness measures beyond expected sampling noise, it is labelled as **alive**. This decision is made via the kill rule in Sections 3.3.1 and 3.3.2.

6. **Adjust test set based on mutation score.** The mutation score is measured by the number of killed mutants (MKR). If a high number of mutants remain alive, the test set is refined by including more diverse variations of sensitive features or adjusting the mutation method to increase fairness sensitivity.

7. **Iterate until fairness adequacy is achieved.** If the mutation score remains low, the process is repeated, beginning from creating new mutants and refining test cases until an adequacy level of fairness is achieved. Once the test set proves to be sufficiently fairness sensitive, the process is considered complete.

3.3 Fairness-Oriented Mutation Testing Framework

3.3.1 Mutation Lifecycle and Kill / Alive Decision

Figure 3.1 summarises the lifecycle of a fairness mutant in the proposed framework. The procedure is model-agnostic: it applies equally to k-NN, Decision Trees, and Logistic Regression, as long as the same model *specification* is retrained on mutated data.

Concretely, for each dataset and base learner, the steps are:

1. Train a *base model* on the original training set.

2. Compute baseline fairness metrics on a fixed test set $\mathcal{S}$.

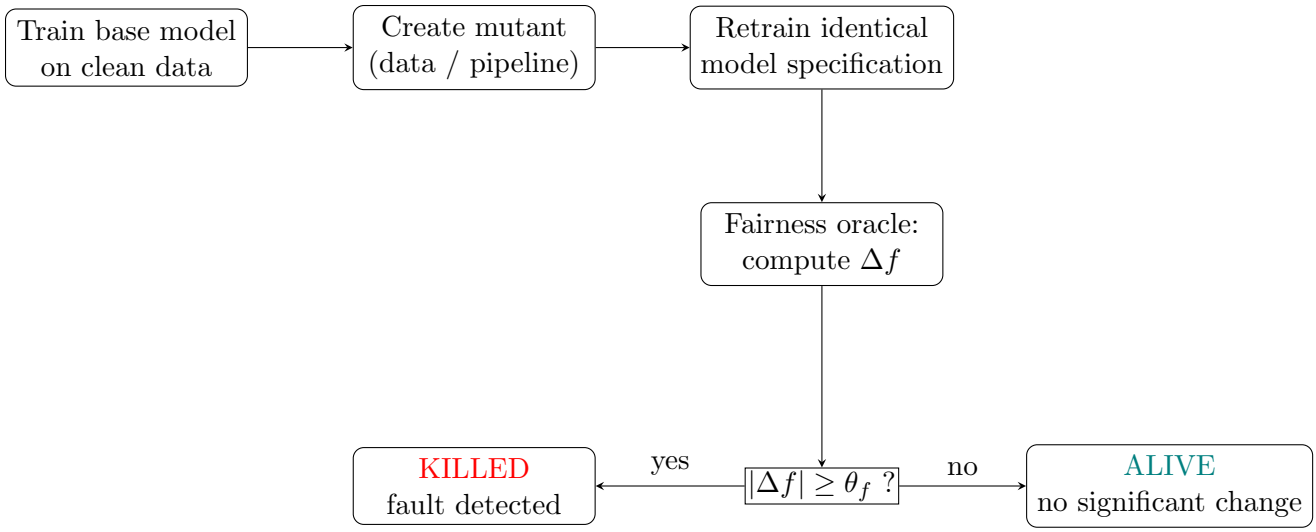

Figure 3.1: Fairness mutation lifecycle. A mutant is *KILLED* if and only if the change in at least one fairness metric exceeds the bootstrap control limit θ_f.

3. Estimate per-metric control limits θ_f using bootstrap (Section 3.3.2).

4. Generate candidate mutants using the taxonomy in Section 3.5.

5. Filter mutants through the validity gate (Section 3.5.6).

6. Retrain the same model specification on each valid mutant and compute fairness metric changes Δf on $\mathcal{S}$.

7. Label each mutant as **KILLED** or **ALIVE** according to the control limit, and aggregate these decisions into MKR (Section 3.6).

3.3.2 Bootstrap Threshold Estimation

To avoid false alarms driven purely by sampling variability, we estimate the *null distribution* of each fairness metric f under *no perturbation*. Let f_{base} denote the value of f for the base model on the test set $\mathcal{S}$.

1. Keep the *base model* fixed.

2. Draw B bootstrap re-samples of the test set, $\{\mathcal{S}^{(b)}\}_{b=1}^{B}$, by sampling with replacement.

3. For each re-sample compute $f_{\text{base}}^{(b)}$ and define $\Delta f^{(b)} = f_{\text{base}}^{(b)} - f_{\text{base}}$.

4. Let μ_{null} and σ_{null} be the mean and standard deviation of $\{\Delta f^{(b)}\}$.

The resulting *control limit* for metric f is

$$\boxed{\theta_f = \mu_{\text{null}} + 1.5\,\sigma_{\text{null}}}$$

where the multiplier 1.5 is chosen as a pragmatic balance between sensitivity and specificity. In our experiments we use $B = 1000$ bootstrap rounds by default.

A mutant m is **KILLED** if

$$|\Delta f_m| \geq \theta_f$$

for *any* monitored fairness metric or subgroup slice; otherwise it is **ALIVE**. Here $\Delta f_m = f_m - f_{\text{base}}$ is the change in the fairness metric for the mutant model relative to the base.

3.4 Fairness Metrics and Headline Score

3.4.1 Group Fairness Metrics

We track three group-fairness criteria (with binary decision $\hat{Y}$, protected attribute A, and groups a, b). These metrics were introduced conceptually in Chapter 2; for completeness we restate their formal definitions here.

1. **Statistical Parity / Demographic Parity.** We measure differences or ratios

of positive outcome rates across groups:

$$\text{SPD} = P(\hat{Y} = 1 \mid A = a) - P(\hat{Y} = 1 \mid A = b),$$

and the corresponding *Demographic Parity Ratio*:

$$\text{DP Ratio} = \frac{\text{Positive Rate}_{\text{privileged}}}{\text{Positive Rate}_{\text{unprivileged}}}.$$

The ideal SPD is 0 and the ideal ratio is 1; in practice, a ratio below 0.8 is often treated as evidence of disparate impact.

2. **Equal Opportunity Difference (EOD).** Equal Opportunity Difference (EOD) measures deviation from equality of opportunity, i.e. the same proportion of each population receives the positive outcome among those who qualify:

$$\text{EOD} = \text{TPR}_{\text{privileged}} - \text{TPR}_{\text{unprivileged}},$$

with $\text{TPR} = P(\hat{Y} = 1 \mid Y = 1, A)$. The ideal value for EOD is 0.

3. **Equalized Odds.** Equalized Odds demands that groups have equal true positive rates (TPR) and false positive rates (FPR):

$$\text{TPR}_{\text{privileged}} = \text{TPR}_{\text{unprivileged}}, \quad \text{FPR}_{\text{privileged}} = \text{FPR}_{\text{unprivileged}}.$$

We operationalise this via the *Equalized Odds Metric* (EOM),

$$\text{EOM} = \max\big\{|\text{TPR}_a - \text{TPR}_b|,\ |\text{FPR}_a - \text{FPR}_b|\big\},$$

with ideal value 0.

Illustrative example. If the blue group has TPR = 0.80 and the red group has TPR = 0.62, then

$$\text{EOD} = 0.80 - 0.62 = 0.18.$$

If the bootstrap-based threshold is $\theta_{\text{EOD}} = 0.10$, the base model already *breaches* the EOD tolerance. In this setting, any mutant that *widens* the TPR gap *beyond 0.10* is treated as a fairness fault and is therefore **KILLED** by the EOD oracle.

3.4.2 Headline Fairness Score

To summarise multiple fairness criteria in a single headline number, we define a Weighted Multi-Objective Fairness Index (WMFI) in $[0, 1]$, where higher is better.

Definition. For each sensitive attribute we compute:

- Demographic Parity gap $\text{DPgap} = \max_g PR_g - \min_g PR_g$,
- Equal Opportunity gap $\text{EOgap} = \max_g TPR_g - \min_g TPR_g$,
- Equalized Odds gap $\text{EODgap} = \max(\text{EOgap}, \max_g FPR_g - \min_g FPR_g)$,
- Disparate Impact $\text{DI} = \min_g PR_g / \max_g PR_g$,

and define

$$\text{WMFI} = 1 - \big(w_{DP}\,\text{DPgap} + w_{EO}\,\text{EOgap} + w_{EOD}\,\text{EODgap}\big) - \lambda \cdot \mathbf{1}[\text{DI} < \tau],$$

with $(w_{DP}, w_{EO}, w_{EOD}) = [0.30, 0.40, 0.30]$, $\tau = 0.80$, and $\lambda = 0.05$.

3.5 Mutation Operators and Mutant Design

Mutation operators apply small changes to the dataset or pipeline to evaluate fairness testing in machine learning models. They are grouped into four main categories: removal operators, addition operators, permutation operators, and shuffling operators. These correspond to the data-level and pipeline-level mutations used in the FAT framework.

3.5.1 Removal Operators

These operators involve the removal of portions of the dataset with the aim of testing fairness on particular data points or features.

Instance Removal

Eliminating particular instances, particularly from minority or protected groups, to simulate data loss:

- **Binary removal.** Remove instances for a sensitive attribute with two values (e.g., gender). For example:
  - Mutant 1: eliminate every case in which gender is female.
  - Mutant 2: eliminate every case in which gender is male.
- **Multi-value removal.** Remove instances involving a multi-valued sensitive attribute (e.g., race). For example:
  - Mutant 1: removal of every case in which race equals Black.
  - Mutant 2: removal of every case in which race equals White.
  - Mutant 3: removal of every case in which race equals Hispanic.

  - Mutant 4: removal of every case in which race equals Asian.

- **Combinatorial removal.** For multiple sensitive attributes, remove instances based on combinations of attribute values. For example:
  - Mutant 1: eliminate all occurrences where gender = male and race = Black.
  - Mutant 2: eliminate all occurrences where gender = male and race = White.
  - Mutant 3: eliminate all occurrences where gender = female and race = Black.
  - Mutant 4: eliminate all occurrences where gender = female and race = White.

Feature Removal

Feature removal deletes characteristics from the data in order to study the model's behaviour without specific types of information.

- **Single feature removal.** Remove one feature entirely from the dataset (e.g., sex). For example:
  - Mutant 1: remove the feature "sex".
- **Multi-feature removal.** Remove multiple features independently from the dataset. For example:
  - Mutant 1: remove "gender".
  - Mutant 2: remove "race".

  - Mutant 3: remove “income”.
  - Mutant 4: remove “education level”.

- **Combinatorial feature removal.** Remove combinations of features from the dataset. For example:
  - Mutant 1: remove “gender” and “race”.
  - Mutant 2: remove “gender” and “income”.
  - Mutant 3: remove “race” and “education level”.
  - Mutant 4: remove “income” and “education level”.

3.5.2 Addition Operators

These operators add new features synthesised from existing data or external sources.

- **Linear combination.** Create a new feature as a linear combination of existing features, such as:

$$\text{New feature} = a \times \text{age} + b \times \text{education years},$$

  where a and b are constants.

- **Ratio.** Create new features based on ratios between attributes, such as:

$$\text{New feature} = \frac{\text{age}}{\text{education years}}.$$

Instance Addition

Generate all possible combinations of sensitive attributes (e.g., gender and race) to create new synthetic instances and test coverage of underrepresented slices.

3.5.3 Permutation Operators

Permutation operators change the order or allocation of data values to test how well the model handles structure in the data.

Feature Permutation

Randomly swap the values of a particular feature between instances to disrupt any pattern the model could be relying on.

Row Permutation

Shuffle the dataset rows to eliminate bias resulting from ordered data (e.g., batch or time series effects).

3.5.4 Shuffling Operators

Shuffling operators randomly reshuffle features or data points to assess how sensitive the model is to input ordering and associations.

Feature Shuffling

Randomly reassign values within each feature column to test for feature dependencies.

Label Shuffling

Randomly shuffle labels among samples to disrupt the direct association between features and labels, testing whether the model learns genuine patterns or simply memorises the data.

3.5.5 Link to Data-, Pipeline-, and Model-Level Mutations

The operators above instantiate the more abstract mutation categories used in the FAT framework:

- **Data-level mutations:** instance/feature removal, addition, permutation, shuffling, label flips, feature noise, intersectional re-sampling, and missingness injection.
- **Pipeline-level mutations:** threshold slide, encoder swap, scaler swap, class-weight changes.
- **Model-level mutations:** regularisation changes and dropping fairness-critical features.

These operators are combined with the validity gate in the next subsection.

3.5.6 Validity Gate (Pre-Filter)

Before a candidate mutant contributes to MKR, it must pass two checks:

1. **Distribution similarity.** We compare the original and mutated training data distributions using two-sample tests:
   - Kolmogorov–Smirnov tests for numeric features;
   - χ^2 tests for categorical features.

   Mutants must satisfy $p \geq 0.05$ (or an effect size below a pre-set ceiling) for all monitored features to be considered within a realistic shift regime.
2. **Training-side fairness shift.** We require that the mutation actually moves the fairness needle on the training set. Let Δf^{train} denote the training-side

change in a fairness metric and $(\mu_{\text{null}}^{\text{train}}, \sigma_{\text{null}}^{\text{train}})$ its bootstrap null statistics on the training data. We require

$$|\Delta f^{\text{train}}| \geq \mu_{\text{null}}^{\text{train}} + \sigma_{\text{null}}^{\text{train}}.$$

Mutants failing either check are discarded (treated as *equivalent* in practice) and do not enter the MKR denominator.

3.6 Mutation Score and Uncertainty

Let M denote the number of mutants that pass the validity gate, and for each such mutant m let $\mathbf{1}\{m \text{ is KILLED}\}$ be an indicator of whether the fairness oracle detects it. The *Fairness Mutation Kill Rate* is then defined as

$$\boxed{\text{MKR} = \frac{\sum_{m=1}^{M} \mathbf{1}\{|\Delta f_m| \geq \theta_f \text{ for some } f\}}{M}}$$

where the condition "for some f" is understood over all monitored metrics and slices. In the computation we exclude mutants deemed equivalent (Section 3.5.6), so M corresponds to non-equivalent mutants only.

To quantify uncertainty, we report 95% percentile bootstrap confidence intervals for MKR by resampling mutants. We also compute slice-wise MKR (e.g., per sensitive attribute or per operator type) to reveal which subgroup, metric, or mutation operator predominantly drives detections. An adjusted score MKR_{adj} further removes mutants with negligible training-side fairness change, making the adequacy assessment more conservative.

3.7 Worked Mini-Example

Consider a concrete example on the Adult dataset, with *sex* as the protected attribute and Logistic Regression as the base learner.

- **Base model:** trained on the original training set with standard pre-processing and regularisation.
- **Mutant:** an *intersectional under-sample* in which females with race $\neq$ White are under-sampled by 20% in the training data.
- **Metric:** Equal Opportunity Difference (EOD) for the sex attribute.

Suppose evaluation on the fixed test set yields:

- $\mathrm{EOD}_{\mathrm{base}} = 0.08$,
- $\mathrm{EOD}_m = 0.19$ for the mutant model, and
- bootstrap-based threshold $\theta_{\mathrm{EOD}} = 0.10$.

The change in EOD is

$$\Delta\mathrm{EOD} = \mathrm{EOD}_m - \mathrm{EOD}_{\mathrm{base}} = 0.19 - 0.08 = 0.11.$$

Since $|\Delta\mathrm{EOD}| = 0.11 \geq 0.10 = \theta_{\mathrm{EOD}}$, the mutant is **KILLED**. It therefore contributes +1 to the MKR numerator. Intuitively, the under-sampling of a vulnerable intersectional subgroup has worsened the TPR gap beyond what could be attributed to noise, and the fairness oracle correctly flags this as a failure.

3.8 Implementation Pseudocode

Algorithm 3 gives high-level pseudocode for the fairness mutation testing pipeline used in this thesis. The same procedure is applied for each dataset and each base learner (k-NN, Decision Tree, Logistic Regression), with the fairness metrics and thresholds shared across mutants.

Algorithm 3: Fairness Mutation Testing Pipeline

Input : Training data $\mathcal{D}_{\text{train}}$; test set $\mathcal{S}$; model template $\mathcal{M}$; mutant generator $\mathcal{G}$; fairness metrics $\mathcal{F}$; threshold multiplier $z = 1.5$; bootstrap rounds $B = 1000$

Output: Mutation Kill Rate (MKR) and 95% confidence interval

`/* Step 1: Bootstrap null distribution for fairness metrics */`

Train base model $M_{\text{base}} \leftarrow \mathcal{M}(\mathcal{D}_{\text{train}})$;

Compute baseline fairness metrics $f_{\text{base}} \leftarrow \text{FairnessOracle}(M_{\text{base}}, \mathcal{S}, \mathcal{F})$;

for $b = 1$ **to** B **do**

  $\mathcal{S}^{(b)} \leftarrow$ bootstrap sample of $\mathcal{S}$;

  $f_{\text{base}}^{(b)} \leftarrow \text{FairnessOracle}(M_{\text{base}}, \mathcal{S}^{(b)}, \mathcal{F})$;

  $\Delta f^{(b)} \leftarrow f_{\text{base}}^{(b)} - f_{\text{base}}$;

end

Compute $\mu_{\text{null}}, \sigma_{\text{null}}$ over $\{\Delta f^{(b)}\}$ for each $f \in \mathcal{F}$;

Set threshold $\theta_f \leftarrow \mu_{\text{null}} + z \cdot \sigma_{\text{null}}$ for each $f \in \mathcal{F}$;

`/* Step 2: Generate, filter, and score mutants */`

$M \leftarrow 0$; kills $\leftarrow 0$;

for *mutant dataset* $\mathcal{D}_{train}^{(m)}$ *in* $\mathcal{G}(\mathcal{D}_{train})$ **do**

  if *validity gate fails for* $\mathcal{D}_{train}^{(m)}$ **then**

    ; `// Discard equivalent or unrealistic mutant`

  end

  $M \leftarrow M + 1$;

  $M_m \leftarrow \mathcal{M}(\mathcal{D}_{\text{train}}^{(m)})$;

  $f_m \leftarrow \text{FairnessOracle}(M_m, \mathcal{S}, \mathcal{F})$;

  $\Delta f_m \leftarrow f_m - f_{\text{base}}$;

  if $\exists f \in \mathcal{F}$ *such that* $|\Delta f_m(f)| \geq \theta_f$ **then**

    kills $\leftarrow$ kills $+1$;

  end

end

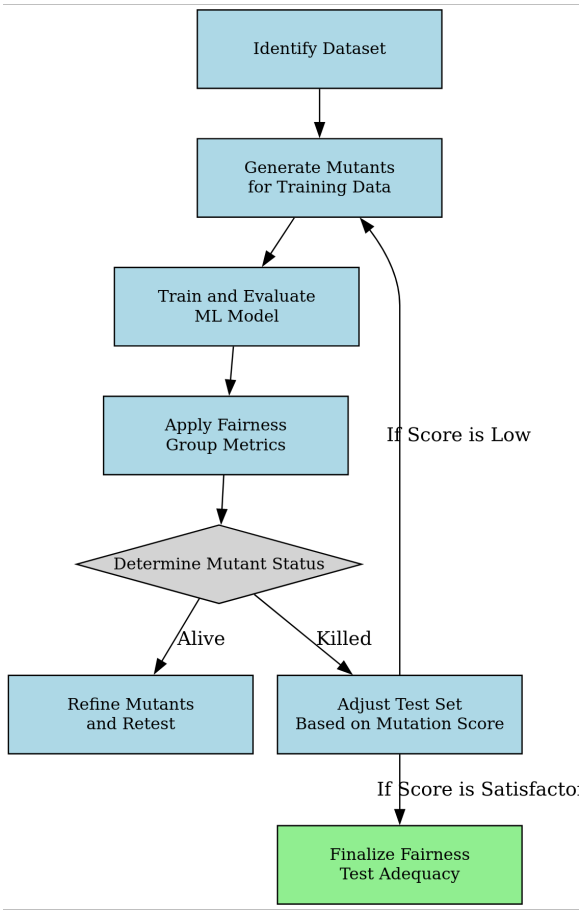

Figure 3.2: Flowchart illustrating the iterative fairness test adequacy process for machine learning models

3.9 Research Questions and Link to Methodology

This chapter operationalises the research questions stated in **??** by:

RQ1. revealing potential biases in machine learning models across sensitive attributes via mutation- and perturbation-based testing (Sections 3.2–3.5);

RQ2. quantifying disparities in model performance across groups using Equal Opportunity Difference (EOD), Equalized Odds Metric (EOM), Demographic Parity, and the WMFI headline score (Section 3.4);

RQ3. specifying and implementing a structured framework for detecting and assessing bias in machine learning systems, and preparing it for empirical validation in the next chapter (Sections 3.3–3.6).

The next chapter instantiates these operators on the Adult and COMPAS datasets, using the base learners described in Chapter 2, and reports empirical MKR and WMFI values to answer these questions.

BIBLIOGRAPHY

[1] Ali, M., Sapiezynski, P., Bogen, M., Korolova, A., Mislove, A., and Rieke, A. Discrimination through optimization: How facebook's ad delivery can lead to skewed outcomes. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW)* (2019).

[2] Angwin, J., Larson, J., Mattu, S., and Kirchner, L. Machine bias. `https://www.proquest.com/newspapers/machine-bias/docview/1828957385/se-2`, May 23 2016. Published by ProPublica.

[3] Barocas, S., Hardt, M., and Narayanan, A. *Fairness and Machine Learning: Limitations and Opportunities*. 2023. Available online at fairmlbook.org.

[4] Behrouz, A., Zhong, P., and Mirrokni, V. Titans: Learning to memorize at test time, 2024.

[5] Bellamy, R. K. E., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., Lohia, P., Martino, J., Mehta, S., Mojsilović, A., Nagar, S., Ramamurthy, K. N., Richards, J., Saha, D., Sattigeri, P., Singh, M., Varshney, K. R., and Zhang, Y. Ai fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development 63*, 4/5 (2019), 4:1–4:15.

[6] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.

[7] Chouldechova, A. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data 5*, 2 (2017), 153–163.

[8] Chouldechova, A. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments, 2017.

[9] Consumer Financial Protection Bureau. Consumer financial protection circular 2022-03: Adverse action notification requirements in credit decisions. `https://www.consumerfinance.gov/`, 2022.

[10] Consumer Financial Protection Bureau. Adverse action notice requirements. `https://www.consumerfinance.gov/`, 2023. Guidance on notices when using AI in credit decisions.

[11] Dastin, J. Amazon scraps secret ai recruiting tool that showed bias against women. *Reuters* (2018).

[12] European Union. Eu artificial intelligence act. `https://eur-lex.europa.eu/`, 2025.

[13] Feldman, M., Friedler, S., Moeller, J., Scheidegger, C., and Venkatasubramanian, S. Certifying and removing disparate impact, 2015.

[14] Galhotra, S., Brun, Y., and Meliou, A. Fairness testing: Testing software for discrimination. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering* (2017), pp. 498–510.

[15] Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., III, H. D., and Crawford, K. Datasheets for datasets. *Commun. ACM 64*, 12 (Nov. 2021), 86–92.

[16] Grote, T. Fairness as adequacy: a sociotechnical view on model evaluation in machine learning. *AI and Ethics 4*, 2 (2024), 427–440.

[17] Hardt, M., Price, E., and Srebro, N. Equality of opportunity in supervised learning, 2016.

[18] Henley, J. Dutch government resigns over child welfare scandal. *The Guardian* (2021).

[19] Himmelreich, J., Hsu, A., Lum, K., and Veomett, E. The intersectionality problem for algorithmic fairness.

[20] Inc., T. Reducing our reliance on image cropping algorithms, 2021.

[21] Jin, D., Liu, X., Liu, Y., Yap, J. Q., Wong, A., Crespo, A., Lin, Q., Yin, Z., Yan, Q., and Ye, R. Infelm: In-depth fairness evaluation of large text-to-image models.

[22] Kearns, M., Neel, S., Roth, A., and Wu, Z. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness.

[23] Lambrecht, A., and Tucker, C. E. Algorithmic bias? an empirical study of apparent gender-based discrimination in the display of stem career ads. *Management Science 65*, 7 (2019), 2966–2981.

[24] Li, J., Zheng, Z., Du, X., Wang, H., and Liu, Y. Drlmutation: A comprehensive framework for mutation testing in deep reinforcement learning systems. *ACM Trans. Softw. Eng. Methodol.* (Mar. 2025).

[25] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. A survey on bias and fairness in machine learning. *ACM Computing Surveys 54*, 6 (2021), 1–35.

[26] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. A survey on bias and fairness in machine learning. *ACM Comput. Surv. 54*, 6 (July 2021).

[27] Mitchell, S., Potash, E., Barocas, S., D'Amour, A., and Lum, K. Algorithmic fairness: Choices, assumptions, and definitions. *Annual Review of Statistics and Its Application 8* (03 2021).

[28] Mobley, D. Mobley v. workday, inc. — court filings and orders. `https://www.courtlistener.com/`, 2025.

[29] Myers, G. J., Sandler, C., and Badgett, T. *The Art of Software Testing*, 3 ed. Wiley, 2011.

[30] Obermeyer, Z., Powers, B., Vogeli, C., and Mullainathan, S. Dissecting racial bias in an algorithm used to manage the health of populations. *Science 366*, 6464 (2019), 447–453.

[31] Obermeyer, Z., Powers, B., Vogeli, C., and Mullainathan, S. Dissecting racial bias in an algorithm used to manage the health of populations. *Science 366*, 6464 (2019), 447–453.

[32] of Representatives, D. H. Ongekend onrecht: Parliamentary interrogation committee on childcare allowance. Tech. rep., Tweede Kamer der Staten-Generaal, January 2021.

[33] Ofqual. Awarding gcse, as and a levels in summer 2020: interim report. Tech. rep., Office of Qualifications and Examinations Regulation, August 2020.

[34] Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J. V., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift, 2019.

[35] Rabanser, S., Günnemann, S., and Lipton, Z. C. Failing loudly: An empirical study of methods for detecting dataset shift.

[36] Sample, I. A-level results: how did the algorithm work and why did it fail? *The Guardian* (2020).

[37] Sharma, A., and Wehrheim, H. Testing machine learning programs. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2020), vol. 12471 LNCS, pp. 5–26.

[38] Tramer, F., Atlidakis, V., Geambasu, R., Hsu, D., Hubaux, J.-P., Humbert, M., Juels, A., and Lin, H. Fairtest: Discovering unwarranted associations in data-driven applications. IEEE, pp. 401–416.

[39] Weerts, H., Dudík, M., Edgar, R., Jalali, A., Lutz, R., and Madaio, M. Fairlearn: Assessing and improving fairness of ai systems. *Journal of machine learning research 24* (2023).