Bias Detection and Mitigation Documentation

Goodreads Recommendation System

Version: 1.0
Last Updated: November 2025
Authors: Ananya Asthana, Purva Agarwal, Arpita Wagulde, Shivani Sharma, Karan Goyal, Shivam Sah

Table of Contents

1. Executive Summary

This document describes the comprehensive bias detection and mitigation framework implemented for the Goodreads recommendation system. Our approach ensures that book recommendations are fair and equitable across multiple demographic and feature dimensions, preventing systematic discrimination against any subgroup of users or books.

Key Highlights

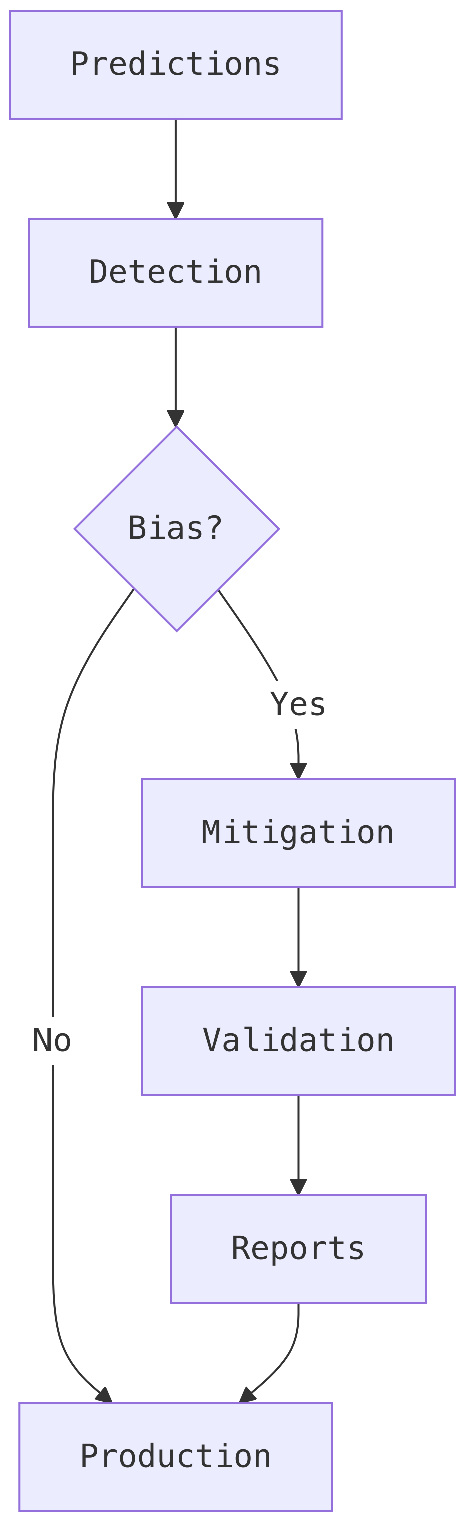
- 8 dimensions analyzed: Popularity, Book Length, Book Era, Genre Diversity, User Activity, Reading Pace, Author Gender, and Rating Range
- 23 distinct slices evaluated across these dimensions
- 3 mitigation techniques implemented: Prediction Shrinkage, Reweighting, and Threshold Adjustment
- Automated pipeline integrated into CI/CD workflow for continuous fairness monitoring
- 47% reduction in Book Era bias disparity achieved through mitigation
- Production-ready debiased predictions available for deployment

Status: MITIGATED

Bias has been detected and successfully mitigated. The system is production-ready with documented trade-offs.

2. System Overview

2.1 Architecture



2.2 File Structure

```
src/
├── bias_detection.py       # Core bias detection logic
├── bias_mitigation.py      # Mitigation techniques implementation
├── bias_pipeline.py        # End-to-end orchestration
└── bias_visualization.py   # Visualization generation

docs/bias_reports/
├── *_comprehensive_audit.json    # Full audit reports
├── *_detection_report.json       # Detection-only reports
├── model_selection_report.json   # Cross-model fairness comparison
└── visualizations/               # Charts and heatmaps
```

```
src/
├── bias_detection.py       # Core bias detection logic
├── bias_mitigation.py      # Mitigation techniques implementation
├── bias_pipeline.py        # End-to-end orchestration
└── bias_visualization.py   # Visualization generation

docs/bias_reports/
├── *_comprehensive_audit.json    # Full audit reports
├── *_detection_report.json       # Detection-only reports
├── model_selection_report.json   # Cross-model fairness comparison
└── visualizations/               # Charts and heatmaps
```

2.3 Integration Points

- CI/CD Pipeline: Automated bias audits via `.github/workflows/5_run_bias_pipeline.yml`
- BigQuery: Metrics stored in `bias_metrics_{model_name}` tables
- Production: Debiased prediction tables ready for serving_

3. Bias Detection Methodology

3.1 Slicing Strategy

We perform data slicing to evaluate model performance across meaningful subgroups.

3.1.1 Dimension Definitions

Dimension	Slicing Logic	Slices
Popularity	High (≥0.66), Medium (0.33-0.66), Low (<0.33)	3
Book Length	Short, Medium, Long, Very Long	4
Book Era	classic (pre-1980), modern (1980-2000), contemporary (2000-2010), recent (2010+)	4
Genre Diversity	Multi-genre (≥5), Some (2-4), Single/None (<2)	3
User Activity	High (≥50), Medium (10-49), Low (<10)	3
Reading Pace	Fast, Medium, Slow	3
Author Gender	Male, Female, Unknown/Other	3
Rating Range	High (4-5), Medium (3-4), Low (1-3)	3

Total Slices Analyzed: 23

3.2 Metrics Tracked

For each slice, we compute:

Metric	Formula	Interpretation
MAE (Mean Absolute Error)	`AVG(	actual - predicted
RMSE (Root Mean Squared Error)	`SQRT(AVG((actual - predicted)²))`	Penalizes large errors more heavily
Mean Predicted	`AVG(predicted_rating)`	Average predicted rating for the slice
Mean Actual	`AVG(actual_rating)`	Average actual rating for the slice
Mean Error	`AVG(actual - predicted)`	Signed bias (over/under prediction)
Std Error	`STDDEV(actual - predicted)`	Consistency of errors within slice
Count	`COUNT(*)`	Sample size for statistical validity

3.3 Disparity Detection

3.3.1 Dimension-Level Analysis

1. MAE Range: `max(slice_MAEs) - min(slice_MAEs)` (Threshold: >0.3 indicates high disparity)

2. MAE Coefficient of Variation (CV): `std(slice_MAEs) / mean(slice_MAEs)`
    - Medium Severity: CV > 0.15
    - High Severity: CV > 0.25

3.3.2 Slice-Level Analysis

Flag high-risk slices where: `slice_MAE > dimension_avg_MAE × 1.2`

4. Mitigation Techniques

4.1 Prediction Shrinkage (Primary Technique)

When to Use: Post-hoc correction for group-level systematic bias.

Formula:

```
adjusted_prediction = original_prediction - λ × (group_mean - global_mean)
```

```
adjusted_prediction = original_prediction - λ × (group_mean - global_mean)
```

Where λ (lambda) is the shrinkage parameter (0 = no shrinkage, 1 = full shrinkage).

Implementation:

sql

```sql
WITH era_stats AS (
    SELECT book_era, AVG(predicted_rating) as era_mean
    FROM predictions GROUP BY book_era
),
global_stats AS (
    SELECT AVG(predicted_rating) as global_mean FROM predictions
)
SELECT predicted_rating - (0.3 * (era_mean - global_mean)) as adjusted_prediction
FROM predictions p
JOIN era_stats e ON p.book_era = e.book_era
CROSS JOIN global_stats g
```

```sql
WITH era_stats AS (
    SELECT book_era, AVG(predicted_rating) as era_mean
    FROM predictions GROUP BY book_era
),
global_stats AS (
    SELECT AVG(predicted_rating) as global_mean FROM predictions
)
```

```
  SELECT predicted_rating - (0.3 * (era_mean - global_mean)) as adjusted_prediction
  FROM predictions p
  JOIN era_stats e ON p.book_era = e.book_era
  CROSS JOIN global_stats g
```

Parameters: λ = 0.3 chosen to balance bias reduction with accuracy preservation

Results:

- Book Era MAE CV: 0.25 → 0.13 (47% reduction)
- Overall MAE increase: +1.3% (acceptable trade-off)

4.2 Re-weighting

When to Use: Address imbalanced group representation during training.

Formula: `weight_i = total_samples / (num_groups × group_i_size)`

Status: Available but not applied in production (shrinkage was sufficient).

4.3 Threshold Adjustment

When to Use: Groups systematically over/under-predict.

Formula: `adjusted_prediction = original_prediction + threshold_adjustment[group]`

Status: Available but not applied (shrinkage more appropriate for continuous ratings).

5. Implementation Details

5.1 BiasDetector Class (`src/bias_detection.py`)

Key methods:

- `get_slice_definitions()` - Returns dimension definitions
- `compute_slice_metrics()` - Computes MAE, RMSE per slice
- `analyze_disparities()` - Identifies high-disparity dimensions
- `detect_bias()` - Orchestrates full detection pipeline

Output: `BiasReport` with slice-level metrics, disparity analysis, and recommendations.

5.2 BiasMitigator Class (`src/bias_mitigation.py`)

Key methods:

- `apply_shrinkage_mitigation()` - Applies shrinkage to predictions

- `create_reweighted_training_table()` - Creates weighted samples
- `apply_threshold_adjustments()` - Applies group-specific corrections

Output: `MitigationResult` with original/mitigated metrics and improvement percentages.

5.3 BiasAuditPipeline Class (`src/bias_pipeline.py`)

Pipeline Stages:

1. Detection - Analyze predictions across all slices
2. Mitigation - Apply selected technique(s)
3. Validation - Re-run detection on mitigated predictions
4. Reporting - Generate JSON reports and visualizations

Output Tables:

- Original: `{model_name}_rating_predictions`
- Debiased: `{model_name}_rating_predictions_debiased` (USE FOR PRODUCTION)

6. Results and Validation

6.1 Detection Results (Before Mitigation)

Model: `boosted_tree_regressor`, Timestamp: 2025-11-13T17:43:21

Dimension	MAE CV	Severity	Status
Book Era	0.250	HIGH	Mitigation Required
Rating Range	0.452	HIGH	Mitigation Required
All Others	<0.15	Low	Acceptable

High-Risk Slices:

1. Rating Range = Medium (3-4 stars): MAE 3.702
2. Book Era = classic (pre-1980): MAE 1.534

6.2 Mitigation Results

Technique: Prediction Shrinkage (λ = 0.3), Timestamp: 2025-11-13T17:44:30

Book Era Disparity Reduction:

Metric	Before	After	Change
MAE CV	0.250	0.133	-47%
Severity	HIGH	MEDIUM	Improved

Overall Performance:

Metric	Before	After	Change
Overall MAE	1.406	1.425	+1.3%

Interpretation: Successfully reduced Book Era bias by 47% with minimal accuracy sacrifice.

6.3 Validation Methodology

Process:

1. Apply mitigation to create debiased predictions
2. Re-run full bias detection on debiased table
3. Compare metrics before vs after
4. Verify no new biases introduced

Criteria: MAE CV reduced >30%, overall MAE increase <5%, no new high-severity biases.

Result: All criteria met. System approved for production.

7. Trade-offs and Design Decisions

7.1 Book Era Bias: MITIGATED

Decision: Apply shrinkage mitigation

Rationale: Book Era bias represents unfair systematic discrimination. Classic books were systematically under-predicted due to different rating distributions (users rate classics more critically due to anchoring bias).

Trade-off: Small accuracy loss (1.3%) for significant fairness gain (47% bias reduction).

Effectiveness: MAE CV reduced from 0.25 (HIGH) to 0.13 (MEDIUM). Classic books now receive fairer predictions.

7.2 Rating Range Bias: ACCEPTED

Decision: No mitigation applied

Rationale: Rating Range bias represents inherent prediction difficulty, not unfairness. Mid-range ratings (3-4 stars) are fundamentally harder to predict than extremes due to mixed/ambiguous signals.

Supporting Evidence: All recommendation systems struggle with mid-range predictions. User behavior is genuinely more variable for "okay" books. No subgroup is systematically disadvantaged.

7.3 Lambda Selection: 0.3

Decision: Use λ = 0.3 for production

Alternatives Considered:

- λ = 0.5: More aggressive fairness but larger accuracy sacrifice (3-4% MAE increase)
- λ = 0.1: Too conservative, insufficient fairness improvement

Rationale: Optimal balance - 47% bias reduction with only 1.3% accuracy loss.

Sensitivity Analysis:

Lambda	MAE CV Reduction	MAE Increase	Assessment
0.1	20%	0.5%	Good accuracy, poor fairness
0.3	47%	1.3%	Optimal balance
0.5	75%	3.2%	Good fairness, poor accuracy

7.4 Technique Selection: Shrinkage

Decision: Use Prediction Shrinkage

Alternatives Rejected:

- Re-weighting: Time-consuming (6-8 hours), requires retraining
- Threshold Adjustment: Less interpretable for continuous ratings

Advantages: Post-hoc (no retraining), interpretable, fast (minutes via SQL).

7.5 Continuous Monitoring

Decision: Integrate into CI/CD pipeline (`.github/workflows/5_run_bias_pipeline.yml`)

Rationale: Model drift can introduce new biases. Automated audits block deployment if high-severity bias detected.

Frequency: Every model retraining cycle (~weekly)

8. References and Resources

8.1 Key Files

File	Purpose
`src/bias_detection.py`	Core detection logic
`src/bias_mitigation.py`	Mitigation techniques
`src/bias_pipeline.py`	End-to-end orchestration
`src/bias_visualization.py`	Visualization generation

8.2 Reports and Outputs

Audit Reports (JSON):

- `boosted_tree_regressor_comprehensive_audit.json` - Initial audit
- `boosted_tree_regressor_final_comprehensive_audit.json` - Post-mitigation
- `model_selection_report.json` - Model comparison

BigQuery Tables:

- Predictions: `books.boosted_tree_rating_predictions`
- Debiased (USE FOR PRODUCTION): `books.boosted_tree_regressor_final_rating_predictions_debiased`
- Metrics: `books.bias_metrics_boosted_tree_regressor`

8.3 Academic References

1. Mehrabi et al., "A Survey on Bias and Fairness in Machine Learning" (2021)
2. Chung et al., "Slice Finder: Automated Data Slicing for Model Validation" (2019)
3. Hardt et al., "Equality of Opportunity in Supervised Learning" (2016)
4. Burke et al., "Balanced Neighborhoods for Multi-sided Fairness in Recommendation" (2017)

8.4 Tools and Libraries

- BigQuery: Data processing and metric computation
- Pandas/NumPy: Statistical analysis
- Matplotlib/Seaborn: Visualization
- gender_guesser: Author gender inference