Compass Analysis using Multi-objective optimization

By: Dhruv Kothari(IMT2022114) and Divyam Sareen(IMT2022010)

1 Introduction & Motivation

1.1 Application Context:

The COMPAS recidivism risk-assessment tool is extensively deployed in U.S. courts to predict the likelihood of defendants reoffending over a two-year horizon. Previous works achieved a good accuracy on Compas dataset, it exhibited large racial disparities in false positive and false negative rates. These disparities are typically quantified via demographic parity difference (DPD), which measures the difference in favorable outcome rates between an unprivileged group and a privileged group. Taking the absolute value of this metric DPD captures the magnitude of bias regardless of its direction. Ensuring low DPD while maintaining high predictive accuracy is therefore critical for equitable, evidence-based decision-making in high-stakes domains.

1.2 Importance and Motivation for Multi-Objective Optimization:

Optimizing exclusively for accuracy can inadvertently worsen fairness violations, resulting in socially unacceptable outcomes. Casting the problem as a multi-objective optimization between accuracy and —DPD— lets us explicitly characterize the trade-off frontier and identify Pareto-optimal models that best balance performance and equity. Indeed, theoretical results prove a fundamental trade-off between statistical parity and accuracy—improving one invariably costs the other. Applying multi-objective optimization approach to the COMPAS dataset produces a spectrum of solutions ranging from highly accurate but biased to fairer but slightly less accurate, offering stakeholders a transparent view of all viable trade-offs. Such a comprehensive trade-off analysis empowers policymakers to select models that meet legal fairness mandates while preserving predictive power.

2 Exploratory Data Analysis & Preprocessing

2.1 Data Loading and Initial Filtering

We begin by loading the COMPAS dataset and applying the same screening-date filters used in prior work to ensure a consistent cohort – namely, keeping only records with $-30 \leq days\_b\_screening\_arrest \leq +30$ to align arrest and screening dates, and removing any entries with missing key fields (e.g., c_jail_in, c_jail_out, decile_score, two_year_recid) – resulting in 6,172 usable records.

2.2 Basic Counts and Class Balance

We first report the overall cohort size, recidivism rate, and racial composition:

- Total Records: 6,172

age_cat	count
25 - 45	3532
Less than 25	1347
Greater than 45	1293

Age Composition

race	count
African-American	3175
Caucasian	2103
Hispanic	509
Other	343
Asian	31
Native American	11

Race Composition

2.3 Feature Distributions and Correlations

- **Age categories** (age_cat) were tallied to verify coverage across bins.
- **Decile score** (the raw COMPAS risk score) was visualized via count-plots for the two largest subgroups—African-American vs. Caucasian.
- We computed length of stay in days from the jail-in/jail-out timestamps and found only a weak positive correlation with decile_score ($r = 0.10$), indicating that time–in–custody is not strongly predictive of score severity.

2.4 Fairness Baseline: Demographic Parity Difference

Before applying multi-objective optimization, we compute the baseline statistical parity difference (DPD) to quantify existing bias in the COMPAS dataset. For a given binary decision threshold α, define

$$\hat{y}_i(\alpha) \;=\; \begin{cases} 1, & \text{if } \dfrac{\text{decile\_score}_i}{\max(\text{decile\_score})} \;\geq\; \alpha, \\ 0, & \text{otherwise.} \end{cases}$$

Let "privileged" be the group with higher baseline positive rate (here, Caucasian) and "unprivileged" its complement. Then the statistical parity difference is

$$\text{DPD}(\alpha) \;=\; P\big(\hat{y} = 1 \mid \text{race} = \text{unprivileged}\big) \;-\; P\big(\hat{y} = 1 \mid \text{race} = \text{privileged}\big).$$

We take the absolute value $|\text{DPD}(\alpha)|$ as our fairness loss, capturing the magnitude of disparity regardless of direction.

3 Bayesian Optimization

3.1 BoTorch

3.1.1 What it is

A flexible, modern library for Bayesian Optimization (BO) built on PyTorch. It's designed for researchers and advanced practitioners who need fine-grained control over BO components.

3.1.2 Working of Bayesian Optimization

1. **Probabilistic Surrogate Model:** BO builds a probabilistic model of the unknown objective function. The most common choice is a Gaussian Process (GP). A GP defines a distribution over functions. Given some observed data points $(x_i, f(x_i))$, the GP can predict the mean and variance (uncertainty) of $f(x)$ at any unobserved point x.
    - *Why PyTorch?* BoTorch leverages PyTorch for automatic differentiation (useful for optimizing acquisition functions and fitting GP hyperparameters) and GPU acceleration.
2. **Acquisition Function:** This function uses the surrogate model's predictions (mean and uncertainty) to decide where to sample next. It balances *exploitation* (sampling where the model predicts a good outcome) and *exploration* (sampling where the model is uncertain, potentially discovering new optima).
    - Common acquisition functions:
        - **Expected Improvement (EI):** Calculates the expected amount of improvement over the current best observed value.
        - **Upper Confidence Bound (UCB):** Chooses points with a high upper confidence bound on their value (mean + weighted standard deviation).
        - **Probability of Improvement (PI):** Chooses points most likely to be better than the current best.
        - **qEI, qUCB, etc.:** Batch versions for selecting multiple points in parallel.
3. **Optimization Loop:**

a) **Initialization:** Evaluate $f(x)$ at a few (e.g., 2–5) initial points, often chosen randomly or via a space-filling design (like Latin Hypercube Sampling).

b) **Fit Surrogate:** Train the GP (or other surrogate model) on all currently observed $(x, f(x))$ pairs. This involves estimating hyperparameters of the GP (e.g., kernel parameters).

c) **Optimize Acquisition Function:** Find the point x_{next} that maximizes the acquisition function. This is itself an optimization problem, but it's usually much cheaper to solve because the acquisition function is analytic and based on the cheap-to-evaluate surrogate model.

d) **Evaluate Objective:** Evaluate the true, expensive black-box function $f(x_{\text{next}})$.

e) **Augment Data:** Add the new pair $(x_{\text{next}}, f(x_{\text{next}}))$ to the dataset.

f) **Repeat:** Go back to step (b) until a budget (e.g., number of evaluations) is exhausted or a satisfactory solution is found.

3.1.3 Key Features & Strengths

- **Flexibility:** Highly customizable models, acquisition functions, and optimization loops.
- **State-of-the-art BO:** Implements many advanced BO techniques (batch optimization, constraints, multi-objective, multi-fidelity, contextual BO).
- **GPU Acceleration:** Benefits from PyTorch for speed.
- **Modular:** Composable components (models, acquisition functions, optimizers).

3.1.4 When to Use

- Expensive black-box optimization (e.g., $<$ 1000s of evaluations).
- When you need advanced BO features or want to research new BO methods.

3.2 Optuna with NSGA-II Sampler

3.2.1 What Optuna is

Optuna is an open-source hyperparameter optimization (HPO) framework designed to automate and streamline the HPO search process. It is recognized for its Pythonic API, ease of integration, and flexibility in supporting various optimization algorithms.

3.2.2 Primary Algorithm(s) Supported by Optuna

Optuna is a versatile framework that provides several built-in and extensible optimization algorithms (samplers). While the Tree-structured Parzen Estimator (TPE) is a common default for single-objective optimization, Optuna's flexibility allows for the use of other samplers, including:

- Random Search
- Grid Search
- CMA-ES (Covariance Matrix Adaptation Evolution Strategy)
- **NSGA-II (Non-dominated Sorting Genetic Algorithm II)**
- Integration with BoTorch/Ax for Gaussian Process-based Bayesian Optimization.
- Quasi-Monte Carlo samplers.

In this project, for each fixed α value, the hyperparameter optimization for the scalarized objective (Equation 8) is conducted using Optuna's `NSGAIISampler`.

3.2.3 Working of NSGA-II (as used with Optuna for each α)

NSGA-II is an evolutionary algorithm. When applied to the single-objective task of minimizing the scalarized fairness-accuracy objective for a specific α, its core mechanisms operate as follows:

1. **Define Objective Function:** A Python function is defined that accepts an Optuna `trial` object. This function uses the `trial` object to:
   - Sample/suggest a set of hyperparameter values for the XGBoost model from their predefined search spaces (e.g., `n_estimators`, `learning_rate`).
   - Train an XGBoost model using these suggested hyperparameters on a validation training split.
   - Evaluate the model on a validation test split to obtain accuracy and DPD.
   - Return a single scalar value representing the objective function (Equation 8) to be minimized.

```
% Example structure (actual project objective is more complex)
def objective(trial):
    # Suggest hyperparameters
    n_estimators = trial.suggest_int("n_estimators", 50, 400)
    learning_rate = trial.suggest_float("learning_rate", 0.005, 0.2, log=True)
    # ... other hyperparameters ...

    # ... train XGBoost model with these params on HPO training data ...
    # ... evaluate on HPO validation data to get acc_val, dpd_val ...

    global current_alpha_for_optuna # Assuming alpha is globally accessible
    score = current_alpha_for_optuna * (1 - acc_val) + \
            (1 - current_alpha_for_optuna) * dpd_val
    return score
```

2. **Study Object and Sampler Configuration:** An Optuna `study` object is created, specifying the optimization 'direction' as ''minimize''. This study is explicitly configured to use the `NSGAIISampler`.

```
sampler = optuna.samplers.NSGAIISampler(seed=20)
study = optuna.create_study(direction='minimize', sampler=sampler)
```

3. **Optimization Loop (Evolutionary Process):** The `study.optimize()` method initiates the HPO process, driven by NSGA-II:
   a) **Initialization:** NSGA-II begins by creating an initial population of candidate solutions (sets of hyperparameters). These can be generated randomly or using a quasi-random sequence if supported by the sampler's initialization. Each solution in the population is evaluated using the 'objective' function.
   b) **Fitness Assignment (Non-dominated Sorting):**
      - Even though we have a single objective for each α-specific HPO run, NSGA-II's core sorting mechanism is based on non-domination. In a single-objective context, this simplifies to sorting the solutions based on their objective function values.
      - Solutions are ranked into different "fronts." In the single-objective case, the first front contains the solutions with the best (lowest) objective values.
   c) **Diversity Preservation (Crowding Distance):** For solutions within the same front (especially relevant if multiple solutions have identical best objective values), a crowding distance metric is calculated. This helps maintain diversity in the population by favoring solutions that are less crowded in the parameter space.
   d) **Selection, Crossover, and Mutation (Generating Offspring):**
      - **Selection:** Parent solutions are selected from the current population, typically favoring those in better (lower-ranked) fronts and those with larger crowding distances. Binary tournament selection is common.

- **Crossover:** Selected parents are combined (e.g., through simulated binary crossover for continuous parameters, or other crossover operators for discrete/categorical parameters) to produce offspring solutions (new sets of hyperparameters).
- **Mutation:** Offspring solutions undergo mutation (e.g., polynomial mutation for continuous parameters, or random changes for discrete parameters) with a certain probability, introducing further exploration and preventing premature convergence.

e) **Evaluate Offspring:** The newly generated offspring (hyperparameter sets) are evaluated using the 'objective' function.

f) **Population Update:** The offspring population is combined with the parent population. This combined population is then sorted again based on non-domination and crowding distance, and the best individuals are selected to form the population for the next generation.

g) **Repeat:** This evolutionary cycle of selection, crossover, mutation, evaluation, and population update is repeated until a predefined termination criterion is met (e.g., a maximum number of trials/evaluations, `N_OPTUNA_TRIALS_PER_ALPHA`).

4. **Pruning (Optional but Supported):** Although NSGA-II itself doesn't inherently define pruning, Optuna's framework allows for the integration of pruning mechanisms. If a pruner is configured with the study, individual trials (evaluations of hyperparameter sets) can be stopped early if they show poor intermediate performance compared to other trials, thus saving computational resources. This is managed by the Optuna study and can work in conjunction with the NSGA-II sampler.

The outcome of this process for each α is the set of hyperparameters from the NSGA-II population that achieved the best (minimum) value for the scalarized objective function.

3.2.4 Key Features & Strengths of Optuna (Relevant to this Usage)

- **Ease of Use:** Pythonic and relatively straightforward API for defining search spaces and objectives.
- **Versatile Sampler Support:** Allows for the selection of various algorithms, including evolutionary ones like NSGA-II.
- **Pruning Capabilities:** Can significantly speed up HPO by terminating unpromising trials early.
- **Parallelization Support:** Optuna studies can be parallelized to leverage multiple cores or machines.
- **Visualization Tools:** Offers utilities for analyzing optimization history, parameter importance, and other aspects of the search process.

3.2.5 Why Use Optuna with NSGA-II

- NSGA-II explores the hyperparameter space to find configurations that minimize the combined accuracy-fairness metric.
- Its population-based nature can be effective in navigating complex search landscapes common in HPO.
- The framework's support for pruning and parallelization remains beneficial.

3.3 USEMO (Uncertainty-aware Search for Expensive Multi-Objective Optimization)

What it is

USEMO is not a specific, off-the-shelf software library like BoTorch or Optuna. Instead, it refers to a *class of algorithms or a conceptual approach* primarily found in academic research, focusing on **expensive multi-objective optimization (MOO)** problems where uncertainty is explicitly managed. Implementations are usually custom-built for specific research.

3.3.1 Primary Algorithm(s)

Typically a hybrid approach combining:

- **Evolutionary Multi-Objective Algorithms (EMOAs):** Such as NSGA-II, NSGA-III, MOEA/D. These algorithms work with a population of solutions and use evolutionary operators (selection, crossover, mutation) to find a set of Pareto-optimal solutions (solutions where no objective can be improved without degrading another).
- **Surrogate Models (often Gaussian Processes):** Similar to BoTorch, surrogate models (one for each objective function) are used to approximate the expensive objective functions and predict their uncertainty.
- **Uncertainty-based Infill Criteria:** Special acquisition functions designed for MOO that guide the selection of new candidate solutions to evaluate. These criteria aim to:
  - Improve the current Pareto front.
  - Reduce uncertainty in promising regions of the Pareto front.
  - Maintain diversity among solutions.
  - Examples: Expected Hypervolume Improvement (EHVI), ParEGO-like approaches, uncertainty-aware dominance checks.

3.3.2 Working

1 **Initialization:** Evaluate an initial population of solutions using the true expensive objective functions.

2 **Fit Surrogate Models:** For each of the k objective functions, train a separate surrogate model (e.g., a GP) based on the evaluated solutions.

3 **Evolutionary Optimization on Surrogates:**

- Run an EMOA (e.g., NSGA-II) for a number of generations. However, instead of evaluating individuals with the *true expensive functions*, evaluate them using the *cheap surrogate models*.
- The selection mechanisms within the EMOA might be modified to be "uncertainty-aware." For example, when comparing two solutions, don't just look at their predicted mean values, but also consider their prediction uncertainty. A solution might be preferred if it has a high chance of dominating another, even if its mean prediction is slightly worse but its uncertainty is high in a favorable direction.

4 **Select Candidate(s) for True Evaluation:** From the population evolved on the surrogates, select one or a few "most promising" individuals to evaluate using the *true expensive functions*. This selection is guided by a multi-objective infill criterion (e.g., EHVI, which selects points expected to maximize the improvement in hypervolume of the dominated space). This step is crucial for balancing exploration and exploitation across multiple objectives.

5 **Evaluate with True Functions:** Evaluate the selected candidate(s) using the actual, expensive objective functions.

6 **Augment Data & Update Surrogates:** Add the newly evaluated solution(s) to the dataset and retrain/update the surrogate models.

7 **Repeat:** Go back to step 3 (or 2 if full retraining is needed) until a budget is exhausted.

3.3.3 Key Features & Strengths

- **Addresses Expensive MOO:** Specifically designed for problems where evaluating objective functions is very costly and multiple conflicting objectives exist.
- **Uncertainty Management:** Explicitly uses the uncertainty from surrogate models to make informed decisions, improving search efficiency.
- **Pareto Front Discovery:** Aims to find a diverse and well-converged set of Pareto-optimal solutions.
- **Sample Efficiency:** Tries to get the best possible Pareto front with the fewest expensive evaluations.

3.3.4 When to Use

- When you have multiple (2+) conflicting objectives that are all expensive to evaluate.
- Requires significant expertise to implement and tune, as it involves combining EMOAs, surrogate modeling, and specialized infill criteria.

3.4 Summary Comparison

Table 1: Summary Comparison of BoTorch, Optuna, and USEMO

Feature	BoTorch	Optuna	USEMO Approach
Primary Goal	Flexible Bayesian Optimization	User-friendly Hyperparameter Optimization (HPO)	Expensive Multi-Objective Optimization (MOO)
Core Algorithm	Bayesian Optimization (GPs, Acq. Func.)	TPE (default), Random, CMA-ES, etc.	Surrogate-Assisted EMOAs (e.g., NSGA-II + GPs)
No. Objectives	Primarily single, good support for multi-objective	Primarily single, some support for multi (NSGA-II)	**Specifically Multi-Objective**
Cost of Eval.	Best for **expensive** evaluations	Handles both cheap and moderately expensive HPO	Best for **very expensive** evaluations
Ease of Use	Moderate to Advanced (research-focused)	**Very Easy** (practitioner-focused)	Advanced (research/custom implementation)
Key Strength	Flexibility, advanced BO, PyTorch integration	Simplicity, pruning, broad HPO algorithm support	Uncertainty handling in expensive MOO

3.5 In Essence

- Use **Optuna** for general hyperparameter optimization due to its ease of use and broad applicability.
- Use **BoTorch** when you need fine-grained control over Bayesian optimization, are dealing with very expensive functions, or require advanced BO features (constraints, batch, multi-objective BO). It can be seen as a powerful backend that Optuna can leverage.
- The **USEMO approach** is relevant when you face truly expensive multi-objective problems and are prepared to delve into more complex, often custom, surrogate-assisted evolutionary algorithms.

Therefore, We decided to use Optuna Solver for our project due to simplicity, easy implementation and decent multi-objective support.

4 Nelder-Mead Search Algorithm

4.1 What it is

The Nelder-Mead algorithm, also known as the downhill simplex method or amoeba method, is a widely used direct search optimization algorithm for finding the minimum (or maximum) of an objective function in a multidimensional space. It is a heuristic search method that does not require gradient information, making it suitable for problems where the objective function is noisy, non-differentiable, or computationally expensive to differentiate.

4.2 Core Concept: The Simplex

The algorithm operates on a geometric figure called a simplex. In an N-dimensional space, a simplex is a polytope formed by $N + 1$ vertices.

- In 1D: A line segment (2 vertices).

- In 2D: A triangle (3 vertices).
- In 3D: A tetrahedron (4 vertices).

Each vertex of the simplex represents a candidate solution (a set of hyperparameter values in our HPO context), and the objective function is evaluated at each vertex.

4.3 Algorithm Steps

The Nelder-Mead algorithm iteratively modifies the simplex to explore the search space and converge towards an optimum. The main steps in each iteration are:

1. **Initialization:**
   - An initial simplex of $N+1$ points (vertices) is created. This can be done by starting with an initial guess x_0 and generating the other N points by adding a step in each dimension, or by other methods.
   - The objective function is evaluated at each of the $N+1$ vertices.
2. **Ordering:** The vertices of the simplex are sorted based on their objective function values, from best (lowest value for minimization) to worst (highest value). Let these be $x_1, x_2, \ldots, x_{N+1}$, where $f(x_1) \leq f(x_2) \leq \cdots \leq f(x_{N+1})$.
   - x_1: Best vertex (current best solution).
   - x_{N+1}: Worst vertex.
   - x_N: Second worst vertex.

   The centroid $\bar{x}$ of all points except the worst vertex x_{N+1} is calculated:
   $$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{1}$$
3. **Transformation Operations:** The algorithm attempts to replace the worst vertex x_{N+1} with a new, better point by applying a sequence of transformations. The standard transformations are:

   a) **Reflection (ρ):** A reflection point x_r is computed by reflecting x_{N+1} through the centroid $\bar{x}$:
   $$x_r = \bar{x} + \rho(\bar{x} - x_{N+1}) \tag{2}$$
   Typically, the reflection coefficient $\rho = 1$. The objective function $f(x_r)$ is evaluated.
   - **Case 1:** $f(x_1) \leq f(x_r) < f(x_N)$: The reflected point x_r is better than the second worst point x_N but not better than the best point x_1. The worst point x_{N+1} is replaced by x_r, and the algorithm proceeds to the next iteration (Ordering).

   b) **Expansion (χ):** If the reflected point x_r is better than the current best point x_1 (i.e., $f(x_r) < f(x_1)$), it indicates that the search is moving in a good direction. An expansion point x_e is computed further along this direction:
   $$x_e = \bar{x} + \chi(x_r - \bar{x}) \tag{3}$$
   Typically, the expansion coefficient $\chi = 2$. The objective function $f(x_e)$ is evaluated.
   - If $f(x_e) < f(x_r)$: The expansion was successful. Replace x_{N+1} with x_e.
   - Else ($f(x_e) \geq f(x_r)$): The expansion failed. Replace x_{N+1} with x_r.

   Proceed to the next iteration (Ordering).

   c) **Contraction (γ):** If the reflected point x_r is worse than or equal to the second worst point x_N (i.e., $f(x_r) \geq f(x_N)$), the simplex might be too large or moving in the wrong direction. A contraction is performed.
   - **Outside Contraction:** If $f(x_r) < f(x_{N+1})$ (reflected point is better than the worst, but not good enough). A contraction point x_c is computed between $\bar{x}$ and x_r:
   $$x_c = \bar{x} + \gamma(x_r - \bar{x}) \tag{4}$$

- **Inside Contraction:** If $f(x_r) \geq f(x_{N+1})$ (reflected point is not better than the worst). A contraction point x_c is computed between $\bar{x}$ and x_{N+1}:

$$x_c = \bar{x} - \gamma(x_{N+1} - \bar{x}) \tag{5}$$

Typically, the contraction coefficient $\gamma = 0.5$. The objective function $f(x_c)$ is evaluated.

- If $f(x_c) < \min(f(x_{N+1}), f(x_r))$: The contraction was successful. Replace x_{N+1} with x_c. Proceed to the next iteration (Ordering).
- Else: The contraction failed. Perform a Shrink operation.

d) **Shrink (σ):** If all above attempts to find a better point than x_{N+1} fail (including contraction), the simplex is shrunk towards the best point x_1. All vertices except x_1 are moved closer to x_1:

$$x_i \leftarrow x_1 + \sigma(x_i - x_1) \quad \text{for } i = 2, \ldots, N+1 \tag{6}$$

Typically, the shrink coefficient $\sigma = 0.5$. The objective function is re-evaluated for all these new N points. Proceed to the next iteration (Ordering).

4. **Termination Criteria:** The iterative process continues until one or more termination criteria are met, such as:

- Maximum number of iterations or function evaluations reached.
- The size of the simplex becomes very small (e.g., the distance between vertices is below a threshold).
- The difference in objective function values at the simplex vertices becomes very small (e.g., $f(x_{N+1}) - f(x_1)$ is below a threshold).

The standard coefficients often used are $\rho = 1, \chi = 2, \gamma = 0.5, \sigma = 0.5$. Some implementations, like SciPy's, might use adaptive parameters.

4.4 Application in Hyperparameter Optimization

In the context of HPO for each α-specific scalarized objective:

- Each **vertex** of the simplex represents a specific configuration of hyperparameters (e.g., `n_estimators`, `learning_rate`, etc.).
- The **objective function value** at a vertex is obtained by training an XGBoost model with that hyperparameter configuration on a validation training split and evaluating the scalarized fairness-accuracy metric (Equation 8) on a validation test split.
- The Nelder-Mead algorithm explores the hyperparameter space by iteratively transforming the simplex, aiming to find the hyperparameter configuration that minimizes this scalarized objective.

4.5 Advantages

- Derivative-free: Does not require gradient information.
- Relatively simple to implement and understand.
- Can be effective for problems with a small number of dimensions (hyperparameters).
- Robust to some level of noise in the objective function.

4.6 Disadvantages

- Can get stuck in local optima, especially for high-dimensional or non-convex problems. The quality of the solution can depend on the initial simplex.
- Performance can degrade in high-dimensional spaces (many hyperparameters), often referred to as the "curse of dimensionality."
- Convergence can be slow, especially if the simplex becomes very elongated or flat.
- Not as theoretically well-understood regarding convergence properties compared to gradient-based methods or some Bayesian optimization techniques.

5 Our Implementation - Bayesian

5.1 Dataset and Preprocessing

The COMPAS dataset (`compas-scores-two-years.csv`) is used. Preprocessing involves:

- **Filtering:** Standard COMPAS filters are applied (e.g., `days_b_screening_arrest` within ± 30 days, valid `is_recid` and `score_text`, non-'O' `c_charge_degree`. The analysis is restricted to African-American and Caucasian individuals for DPD calculation.
- **Feature Selection:** Features such as sex, age, juvenile and prior crime counts, charge degree, and days between screening and arrest are used. The target is `two_year_recid`.
- **Missing Value Imputation:** Median imputation is applied to `days_b_screening_arrest`.
- **Encoding and Scaling:** Numerical features are standardized using `StandardScaler`. Categorical features (including 'race') are one-hot encoded using `OneHotEncoder` (with drop='first').
- **Data Splitting:** The data is split into an 80% training set and a 20% test set, stratified by the target variable. The original 'race' labels from the test set are preserved for DPD calculation.

The `load_and_preprocess_data()` function encapsulates these steps.

5.2 Metrics

- **Accuracy:** Standard classification accuracy, calculated as $\frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$.
- **Demographic Parity Difference (DPD):** Measures the absolute difference in the positive prediction rates (PPR) between the unprivileged group (African-American) and the privileged group (Caucasian):

$$\text{DPD} = |\text{PPR}_{\text{unprivileged}} - \text{PPR}_{\text{privileged}}| \tag{7}$$

  where PPR is the proportion of individuals within a group for whom a positive outcome (recidivism) is predicted. A DPD of 0 indicates perfect demographic parity. This is implemented in the `demographic_parity_difference()` function.

5.3 Scalarized Objective Function for Optimization

To find trade-off solutions, a scalarized objective function is minimized:

$$\text{Objective} = \alpha \cdot (1 - \text{Accuracy}) + (1 - \alpha) \cdot \text{DPD} \tag{8}$$

Here, $(1 - \text{Accuracy})$ is the error rate. The parameter $\alpha \in [0, 1]$ controls the trade-off:

- $\alpha = 0$: Minimizes DPD only.
- $\alpha = 1$: Minimizes error rate (maximizes accuracy) only.
- $0 < \alpha < 1$: Balances both objectives.

5.4 Hyperparameter Optimization (HPO) with Optuna

For each value of α, Optuna is used to find the optimal hyperparameters for an XGBoost classifier that minimize Equation 8.

- **Optuna 'objective()' function:** This function takes an Optuna 'trial' object. It suggests XGBoost hyperparameters (e.g., `n_estimators`, `learning_rate`, `max_depth`) using `trial.suggest_*` methods. It then trains an XGBoost model on a dedicated validation split (derived from the main training set) and evaluates its accuracy and DPD on this split. The scalarized objective value (Equation 8) is returned.
- **Optuna 'study':** An Optuna 'study' object is created for each α, configured to use the 'NSGAIISampler' (typically used for multi-objective). The 'study.optimize()' method runs the HPO for a specified number of trials.

5.5 Pareto Frontier Generation (`find_pareto_frontier()`)

This function orchestrates the entire process:

1. **Initialization:** Starts with $\alpha = 0.0$ and $\alpha = 1.0$ in a queue (`alphas_to_evaluate`). A set `processed_alphas` tracks evaluated α values, and `candidate_pareto_points` stores `(accuracy, DPD, alpha, best_params)` tuples.
2. **Iterative Alpha Evaluation:** The main loop continues until a target number of α values (`max_points`) have been evaluated or the queue is empty.
   - For each α from the queue:
     (a) Perform HPO using Optuna (as described above) to find `best_hyperparams`.
     (b) Train a final XGBoost model on the full training set (`X_train_full_processed`) using these `best_hyperparams`.
     (c) Evaluate this model on the held-out test set to obtain final `acc_test` and `dpd_test`.
     (d) Store the result in `candidate_pareto_points`.
3. **Adaptive Weight Generation:** After evaluating an α (and if more points are desired and at least two candidates exist):
   (a) Filter current `candidate_pareto_points` to get the non-dominated set.
   (b) If at least two non-dominated points exist, sort them by DPD.
   (c) Calculate the Euclidean distance in the (accuracy, DPD) objective space between all consecutive non-dominated points.
   (d) **Primary Midpoint Strategy:** Identify the pair of points with the largest distance. A new α_{new} is calculated as the average of the α values corresponding to these two points. If α_{new} is unique (not in `processed_alphas` or `alphas_to_evaluate`) and not excessively close to existing alphas, it's added to the `alphas_to_evaluate` queue.
   (e) **Backup Random Strategy:** If the midpoint strategy fails to generate a new, unique α, a random α is generated within the range defined by the α values of the point pair exhibiting the largest distance. This also undergoes uniqueness and proximity checks before being added to the queue.

5.6 Filtering Non-Dominated Solutions (`filter_non_dominated()`)

After all α values are processed, the `candidate_pareto_points` list may contain dominated solutions. This function filters them out:

- A point $P_i = (\text{accuracy}_i, \text{DPD}_i)$ is dominated by $P_j = (\text{accuracy}_j, \text{DPD}_j)$ if:
  1. $\text{accuracy}_j \geq \text{accuracy}_i$ AND $\text{DPD}_j \leq \text{DPD}_i$, AND
  2. ($\text{accuracy}_j > \text{accuracy}_i$ OR $\text{DPD}_j < \text{DPD}_i$).
- Only non-dominated points are retained to form the final Pareto frontier.

6 Results and Discussion - Bayesian

6.1 Generated Pareto Frontier

The hyperparameter optimization process, guided by varying the trade-off parameter α, yielded a set of candidate solutions. After filtering for non-dominated points, the resulting Pareto frontier is depicted in Figure 2.

6.2 Inferences from the Pareto Frontier

The generated Pareto frontier (Figure 2) provides several key insights into the accuracy-fairness trade-off for this specific problem setup:

1. **Existence of a Trade-off:** The characteristic shape of the frontier, where improving one objective (e.g., increasing accuracy) generally leads to a worsening of the other (e.g., increasing DPD), clearly demonstrates the inherent trade-off between accuracy and demographic parity. It is generally not possible to simultaneously achieve the highest accuracy and the lowest DPD (highest fairness).

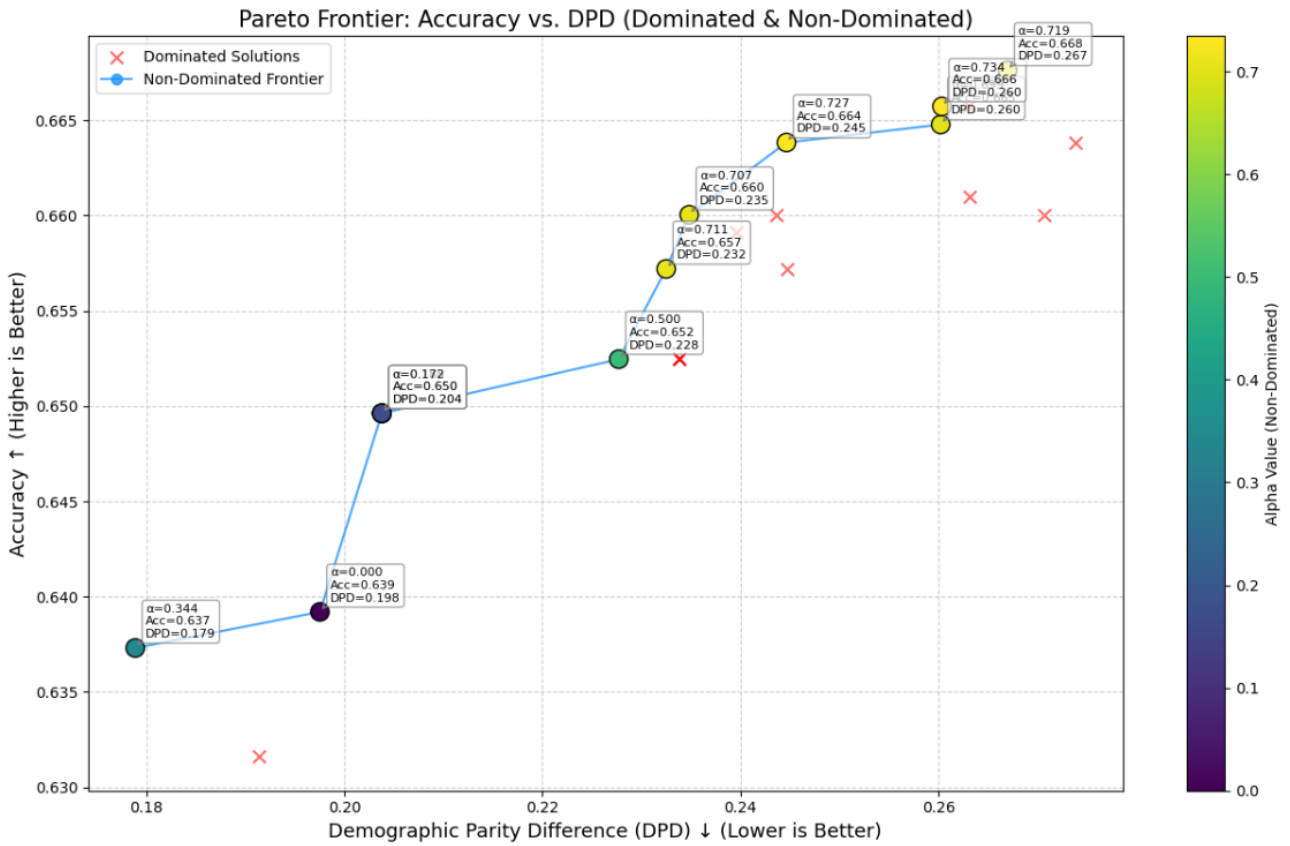

Figure 2: Pareto Frontier: Accuracy vs. Demographic Parity Difference (DPD) for COMPAS Recidivism Prediction. Non-dominated solutions are connected by a line and colored by their corresponding α value. Dominated solutions are marked with red 'x's.

2. **Range of Optimal Solutions:** The frontier presents a spectrum of optimal models.
   - At the lower-left end (e.g., $\alpha \approx 0.344$ resulting in DPD ≈ 0.179 and Accuracy ≈ 0.637), models prioritize fairness, achieving a lower DPD but at the cost of lower predictive accuracy. The point with $\alpha = 0.000$ (DPD ≈ 0.198, Acc ≈ 0.639) represents the model most focused on minimizing DPD.
   - Conversely, at the upper-right end (e.g., $\alpha \approx 0.719$ to $\alpha \approx 0.734$, with DPDs around $0.260 - 0.267$ and Accuracies around $0.666 - 0.668$), models prioritize accuracy, achieving higher predictive performance but exhibiting a larger disparity in prediction rates between groups.
3. **Impact of the Trade-off Parameter (α):** The color gradient of the non-dominated points, corresponding to the α value, illustrates how the optimization shifts focus.
   - Lower α values (cooler colors like dark purple/blue, e.g., $\alpha = 0.000, 0.172, 0.344$) lead to solutions on the lower-DPD, lower-accuracy part of the frontier.
   - Higher α values (warmer colors like yellow, e.g., $\alpha = 0.707, 0.711, 0.727, 0.734$) yield solutions on the higher-accuracy, higher-DPD part of the frontier.
   - The point with $\alpha = 0.500$ (Acc ≈ 0.652, DPD ≈ 0.228) represents a model where accuracy and DPD were given roughly equal importance in the scalarized objective during its HPO.
4. **Diminishing Returns / "Knee" Points:** While not extremely sharp in this particular graph, Pareto frontiers often exhibit "knee" points where a small sacrifice in one objective can lead to a significant gain in the other. For instance, moving from the point at $\alpha = 0.500$ (Acc ≈ 0.652, DPD ≈ 0.228) to $\alpha = 0.711$ (Acc ≈ 0.657, DPD ≈ 0.232) yields a small accuracy gain for a small increase in DPD. However, further pursuing higher accuracy (e.g., towards $\alpha = 0.734$) might lead to more rapidly increasing DPD for smaller accuracy gains. The region around $\alpha = 0.172$ to $\alpha = 0.500$ shows a noticeable slope change.
5. **Dominated Solutions:** The red 'x' marks represent hyperparameter configurations that were evaluated but are sub-optimal. For any dominated solution, there is at least one point on the Pareto frontier that is better

or equal in both accuracy and DPD, and strictly better in at least one. This highlights the effectiveness of the optimization process in identifying superior trade-offs. For example, the dominated point near (DPD=0.23, Acc=0.652) is clearly inferior to the non-dominated point with $\alpha = 0.500$ at almost the same DPD but higher accuracy.

6. **Guidance for Model Selection:** This frontier does not dictate a single "best" model. Instead, it provides a set of optimal choices. The selection of a specific model from this frontier would depend on the stakeholder's relative valuation of accuracy versus fairness for the particular application context. For instance, if a DPD above 0.25 is deemed unacceptable, solutions on the right side of the frontier would be excluded, guiding the selection towards models with higher fairness, albeit potentially lower accuracy.

7. **Effectiveness of Adaptive Alpha:** The spread of points along the frontier, generated by varying α values (including those from the adaptive strategy), suggests that the adaptive weighting helped in exploring different regions of the trade-off space.

7 Results and Discussion - Nelder Meads

This section presents the empirical results obtained from applying the Pareto optimization methodology, utilizing the Nelder-Mead search algorithm for hyperparameter optimization, to the COMPAS dataset. The primary output is the Pareto frontier, which visualizes the optimal trade-offs between predictive accuracy and Demographic Parity Difference (DPD).

7.1 Generated Pareto Frontier with Nelder-Mead

The hyperparameter optimization process for each α value, guided by the Nelder-Mead algorithm, yielded a set of candidate solutions. After filtering for non-dominated points, the resulting Pareto frontier is depicted in Figure 3.

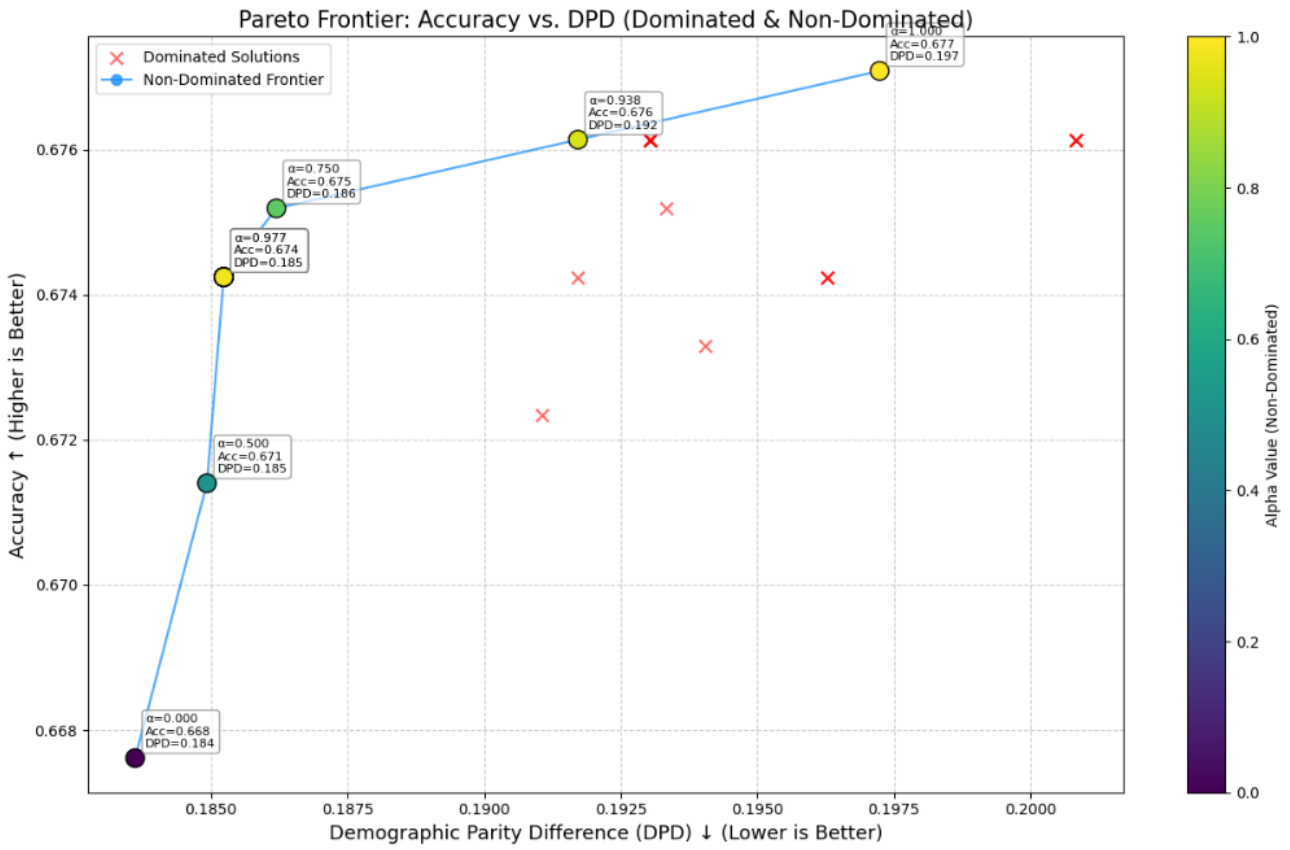

Figure 3: Pareto Frontier using Nelder-Mead HPO: Accuracy vs. Demographic Parity Difference (DPD) for COMPAS Recidivism Prediction. Non-dominated solutions are connected by a line and colored by their corresponding α value. Dominated solutions are marked with red 'x's.

7.2 Inferences from the Pareto Frontier (Nelder-Mead)

The Pareto frontier generated using Nelder-Mead for hyperparameter optimization (Figure 3) reveals several characteristics of the accuracy-fairness trade-off:

1. **Clear Trade-off Visualization:** Similar to previous observations, the graph clearly illustrates the fundamental trade-off. As models aim for higher accuracy (moving upwards), there is a general tendency for the Demographic Parity Difference to increase (moving rightwards, indicating reduced fairness). Conversely, prioritizing lower DPD often comes at the expense of some predictive accuracy.
2. **Range and Nature of Optimal Solutions:** The frontier spans a range of DPD from approximately 0.184 to 0.197, with corresponding accuracies from about 0.668 to 0.677.
    - The point associated with $\alpha = 0.000$ (Acc ≈ 0.668, DPD ≈ 0.184) represents the solution most focused on minimizing DPD. This solution achieves the best fairness on the frontier but has the lowest accuracy among the non-dominated points.
    - The point associated with $\alpha = 1.000$ (Acc ≈ 0.677, DPD ≈ 0.197) represents the solution most focused on maximizing accuracy (minimizing error rate). It achieves the highest accuracy on the frontier but also exhibits the highest DPD.
3. **Influence of the Trade-off Parameter (α):** The color mapping (from dark purple/blue for low α to yellow for high α) effectively shows how the optimization's focus shifts along the frontier.
    - Low α values (e.g., $\alpha = 0.000$) produce solutions in the lower-left region, prioritizing fairness (low DPD).
    - Intermediate α values (e.g., $\alpha = 0.500$ with Acc ≈ 0.671, DPD ≈ 0.185; $\alpha = 0.750$ with Acc ≈ 0.675, DPD ≈ 0.186) find balanced solutions.
    - High α values (e.g., $\alpha = 0.938, 0.977, 1.000$) push the solutions towards the upper-right, prioritizing accuracy.
4. **Shape of the Frontier and Sensitivity:** The frontier in this graph appears relatively "flat" in its initial segment (from $\alpha = 0.000$ to $\alpha = 0.750$), where significant increases in α (and thus emphasis on accuracy) lead to notable gains in accuracy with only very modest increases in DPD. For example, moving from $\alpha = 0.000$ (DPD ≈ 0.184) to $\alpha = 0.750$ (DPD ≈ 0.186) increases accuracy from ≈ 0.668 to ≈ 0.675 with only a slight DPD increase. However, pushing for the very highest accuracies (e.g., from $\alpha = 0.938$ to $\alpha = 1.000$) results in a more pronounced increase in DPD (from ≈ 0.192 to ≈ 0.197) for a smaller gain in accuracy (from ≈ 0.676 to ≈ 0.677). This suggests a region where the cost of fairness for marginal accuracy gains becomes steeper.
5. **Dominated Solutions:** The presence of several red 'x' marks (dominated solutions) indicates that the Nelder-Mead search, combined with the adaptive alpha strategy, explored various hyperparameter configurations, many of which were sub-optimal compared to those on the discovered frontier. This reinforces the value of the Pareto optimization approach in identifying the most efficient trade-offs.
6. **Decision-Making Utility:** The frontier provides a clear set of trade-off options. For instance, a stakeholder might observe that an accuracy of around 0.675 can be achieved with a DPD of 0.186 ($\alpha = 0.750$). If slightly higher accuracy is desired, say 0.676 ($\alpha = 0.938$), it comes with an increase in DPD to 0.192. The "cost" of this extra 0.001 accuracy is an increase of 0.006 in DPD. This quantitative insight is crucial for informed decision-making based on organizational priorities regarding fairness and performance.

8 Conclusion

This study explored recidivism prediction on the COMPAS dataset through a multi-objective lens, balancing predictive accuracy against fairness (Demographic Parity Difference, DPD). We implemented two optimization strategies—Optuna's NSGA-II sampler and the derivative-free Nelder-Mead algorithm—across a range of trade-off weights (α) to generate Pareto frontiers of non-dominated models. Both methods revealed the fundamental tension: reducing racial disparity in positive predictions necessitates a slight drop in accuracy, while pursuing maximum accuracy increases DPD. Optuna's NSGA-II produced a smoothly distributed frontier, highlighting nuanced "knee" regions for informed model selection, whereas Nelder-Mead yielded comparable trade-off ranges with fewer but distinct optimal points. Future work may extend this framework to other fairness criteria (e.g., equalized odds), incorporate cost-sensitive loss functions, or explore surrogate-assisted evolutionary algorithms to enhance search efficiency.