VirnyFlow: A Design Space for Responsible Model Development [Scalable Data Science]

Denys Herasymuk
Ukrainian Catholic University
Lviv, Ukraine
herasymuk@ucu.edu.ua

Nazar Protsiv
Ukrainian Catholic University
Lviv, Ukraine
protsiv.pn@ucu.edu.ua

Julia Stoyanovich
New York University
New York, USA
stoyanovich@nyu.edu

ABSTRACT

Developing machine learning (ML) models requires a deep understanding of real-world problems, which are inherently multi-objective. In this paper, we present VirnyFlow, the first design space for responsible model development, designed to assist data scientists in building ML pipelines that are tailored to the specific context of their problem. Unlike conventional AutoML frameworks, VirnyFlow enables users to define customized optimization criteria, perform comprehensive experimentation across pipeline stages, and iteratively refine models in alignment with real-world constraints. Our system integrates evaluation protocol definition, multi-objective Bayesian optimization, cost-aware multi-armed bandits, query optimization, and distributed parallelism into a unified architecture. We show that VirnyFlow significantly outperforms state-of-the-art AutoML systems in both optimization quality and scalability across five real-world benchmarks, offering a flexible, efficient, and responsible alternative to black-box automation in ML development.

PVLDB Reference Format:
Denys Herasymuk, Nazar Protsiv, and Julia Stoyanovich. VirnyFlow: A Design Space for Responsible Model Development [Scalable Data Science]. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/denysgerasymuk799/virny-flow.

1 INTRODUCTION

Developing machine learning (ML) models responsibly requires a deep understanding of real-world problems, which are inherently multi-objective. Responsible model development extends beyond optimizing for accuracy [37], requiring an *evaluation protocol* tailored to the specific context of use and guided by human expertise [76]. At the same time, constructing a well-suited ML pipeline demands extensive experimentation, iterative refinements, and significant computational resources. Ideally, systems designed to support model developers should follow a human-centric approach, offering a diverse set of pipeline tuning criteria, enabling multi-stage pipeline optimization, and providing an efficient and flexible *design space* for comprehensive experimentation.

As a practical scenario, consider Ann, a data scientist working on a public policy task, such as to predict whether a low-income individual is eligible for public health insurance (as in ACSPublicCoverage [23]). Ann aims to build an accurate, robust, and fair ML pipeline, which presents several challenges. First, she must encode multiple performance dimensions—fairness across sex, race, their intersections, and stability—into the objective. Optimizing one metric rarely improves others [4, 19, 44, 77] because of non-convex fairness constraints and tuning issues [77]. Second, objectives must span the full ML lifecycle: errors introduced during data collection can propagate through preprocessing (e.g., imputation), model selection, and tuning [4, 66, 67, 77]. Third, efficiency matters: exploring *tens of thousands* of pipeline variants demands distributed resources and early pruning. Finally, large-scale execution remains complex when Ann revises objectives or shifts from unconstrained multi-objective to constrained single-objective search.

The system we describe in this paper, `VirnyFlow`, will assist Ann in her task. We preview the system's flexibility in Listing 1 that shows an experiment config Ann may specify for her scenario.

```
pipeline_args:
  dataset: "folk_pubcov"
  sensitive_attrs_for_intervention: ["SEX", "RAC1P"]
  null_imputers: ["median-mode", "miss_forest", "datawig"]
  fairness_interventions: ["DIR", "AD"]
  models: ["lr_clf", "rf_clf", "lgbm_clf", "gandalf_clf"]

optimisation_args:
  ref_point: [0.40, 0.10, 0.10]
  objectives:
   - {name: "obj_1", metric: "F1", group: "overall", weight: 0.25}
   - {name: "obj_2", metric: "SRD", group: "SEX&RAC1P", weight: 0.5}
   - {name: "obj_3", metric: "Label_Stability", group: "overall",
      weight: 0.25}
  max_total_pipelines_num: 100
  num_workers: 32
  num_pp_candidates: 4
  training_set_fractions_for_halting: [0.5, 1.0]
  exploration_factor: 0.5
  risk_factor: 0.5

virny_args:
  bootstrap_fraction: 0.8
  n_estimators: 50
  sensitive_attrs: {SEX:'2', RAC1P:['2','3','4','5','6','7','8','9'],
                    SEX&RAC1P: None}
```

Listing 1: Experiment config example: Multi-objective optimization with model selection, fairness interventions, and null imputation.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

1.1 Background and Related Work

ML pipeline optimization has produced a diverse ecosystem of *AutoML* tools that fall into three broad families [7]. *Hyperparameter-optimization* frameworks treat the pipeline as fixed and search only its hyperparameters. *Neural-architecture search* (NAS) enlarges the space by optimizing both hyperparameters and the topology of deep networks. Finally, *broad-spectrum AutoML* tackles the full *Combined Algorithm Selection and Hyperparameter Optimization (CASH)* problem [73], exploring alternative algorithms, preprocessors, and hyperparameters with techniques such as Bayesian optimization (`Auto-sklearn` [29], `AutoGluon` [26]), bandits (`ATM` [69], `Alpine Meadow` [65]), evolutionary search (`TPOT` [56], `FEDOT` [54]), and Monte-Carlo tree search (`MOSAIC` [61], `Oracle AutoML` [80]).

Despite their sophistication, most AutoML systems define optimization criteria in isolation from the problem context. This omission is risky: studies show that ML models can reproduce, amplify, or introduce bias, harming minority groups [14, 17, 22, 55, 70, 72]. Unless *fairness* is an explicit objective, AutoML may deepen existing disparities. Yet fairness itself cannot be fully automated [76]; deciding what is fair depends on the socio-technical setting and requires human judgment. Domain experts therefore must choose appropriate fairness metrics and decide how to embed them in the optimization loop [79]. Beyond fairness, other performance dimensions such as *model stability* also play a critical role [5]. In high-stakes domains like healthcare, an unstable model can lead to inconsistent predictions and potentially harmful consequences for individuals and institutions [48]. Optimization processes that ignore stability risk producing models that are unreliable in practice, even if they appear accurate in evaluation settings.

Meaningfully incorporating fairness, stability, or other performance criteria into model development requires optimization beyond defining an *evaluation protocol*, spanning the *entire ML lifecycle* [66, 67, 77]. Model performance is heavily influenced by data quality and pipeline design choices [63]; biases originating from data collection (e.g., imbalanced sampling [28, 81]) propagate downstream. Suboptimal decisions during *pre-processing* [4, 34, 63], *hyperparameter tuning* [77], or *model selection* [37] further distort outcomes, underscoring the need to support *multi-stage, multi-objective* optimization from data preparation through fairness interventions.

While recent efforts in AutoML have aimed to incorporate fairness into the optimization process [21, 24, 46, 53, 59, 62, 78], these *fairness-aware AutoML* frameworks still face critical limitations in addressing real-world problems. Most notably, they often operate only with binary (non-intersectional) groups and rely on a limited set of fairness metrics, failing to capture the broader range of desiderata [9]. Additionally, these systems fail to account for such crucial dimensions of model performance as stability and uncertainty [13, 27, 31, 37]. Finally, they typically focus only on model tuning and selection, and do not consider multiple lifecycle stages.

In summary, despite significant advances in AutoML, no existing system fully supports a context-sensitive, iterative ML pipeline development guided by human domain expertise, as emphasized by Weerts et al. [76]. `Alpine Meadow` [65] comes closest, particularly in its focus on interactivity and human-centered design, and we adopt two key ideas from it. *First*, we combine multi-armed bandits with Bayesian optimization (BO) to improve exploration and interactivity; however, unlike `Alpine Meadow`, `VirnyFlow` integrates a comprehensive *evaluation protocol* throughout the architecture, allowing flexible definition of optimization objectives, multi-dimensional model measurement, and multi-objective BO during tuning. *Second*, we use similar query optimization techniques and pruning strategies based on `Alpine Meadow`'s *adaptive pipeline selection* (Algorithm 1), extending both scoring and pruning methods to the multi-objective setting.

We acknowledge that selecting appropriate evaluation metrics can be challenging and therefore requires iterative refinement and experimentation. To address this, `VirnyFlow` incorporates distributed parallelism into its architecture to enhance scalability, efficiency, and resource utilization.

1.2 Scope and Contributions

In this paper, we present `VirnyFlow`, the first *design space* for responsible ML model development, assisting data scientists in constructing pipelines tailored to their problem context. We avoid referring to `VirnyFlow` as an AutoML system, as it is not an autonomous black-box optimizer outputting a single "best" pipeline without considering contextual factors. Instead, `VirnyFlow` provides a flexible, interactive environment supporting rapid iteration, extensive experimentation, and context-specific optimization criteria, enabling users to shape pipelines to meet their needs.

Grounded in human-centric design, `VirnyFlow` offers four essential features: *(i)* flexible definition of optimization objectives; *(ii)* multi-stage, multi-objective pipeline optimization with customization; *(iii)* comprehensive, interactive experiment management with query optimization; and *(iv)* efficient distributed parallelism. Our main contributions are:

- **System architecture.** We present a novel architecture tailored to responsible ML pipeline development.
- **Evaluation protocol integration.** We define and embed a context-sensitive *evaluation protocol* into the architecture to support multi-stage, multi-objective optimization across the ML lifecycle. This includes tuning for fairness and stability alongside accuracy, with optimization criteria defined over flexible data subsets (e.g., demographic groups and intersections).
- **Unified optimization framework.** We combine evaluation protocol specification, multi-objective Bayesian optimization, cost-aware bandits, query optimization, and distributed parallelism into a cohesive *design space* for iterative experimentation.
- **Empirical validation.** We show that `VirnyFlow` outperforms state-of-the-art AutoML systems in both optimization flexibility and computational scalability on five real-world benchmarks.

For reasons of scope, we focus on settings characterized by moderately sized tabular datasets and diverse pipeline variants, rather than large-scale datasets or distributed training of complex models. We emphasize pipeline execution optimizations, excluding visual integration, user feedback, or interface design. Additionally, we restrict our consideration to traditional supervised ML pipelines with fixed structures, omitting joint data cleaning and training, neural architecture search, unsupervised learning, and automated data acquisition or preprocessing. While `VirnyFlow` is extensible to these broader tasks, exploring them is beyond this paper's scope.

2 SYSTEM DESIGN

This section presents the system design of `VirnyFlow`, which supports flexible performance criteria, multi-stage and multi-objective optimization, scalable execution, and comprehensive experiment management. We begin with an overview of the optimization process, then describe each step in detail, and finish by explaining how `VirnyFlow` enables distributed execution.

2.1 The Optimization Process

Figure 1 shows the key steps of the `VirnyFlow` optimization process. Steps 1–3 take place on the coordinator node, while steps 4 and 5 execute on the worker nodes.

(1) Search space construction: The execution begins when a user provides an *experiment config* to `Task Manager`, similar to one in Listing 1, and defines a hyper-parameter search space for each stage of the pipeline. This configuration specifies pipeline stages (e.g., null imputation, fairness intervention, or model evaluation), each with multiple variants, including user-defined custom components. `Task Manager` then constructs a search space of *logical pipelines*—directed acyclic graphs (DAGs) of primitives with their hyper-parameter domain specification (not fixed).

(2) Logical pipeline selection: To efficiently explore the search space, `Task Manager` uses a cost model that prioritizes promising pipelines based on prior results.

(3) Physical pipeline selection: Selected *logical pipelines* are instantiated into *k physical pipelines*, with random candidates added to help avoid local optima. *Physical pipelines* are derived from a *logical pipeline* using Bayesian optimization (BO).

(4) Pipeline evaluation: `Workers` execute and prune the selected *physical pipelines*. `VirnyFlow` uses the *Adaptive Pipeline Selection* algorithm from `Alpine Meadow`, reducing training time by allocating resources adaptively.

(5) Iterative refinement: Pipeline evaluation results are stored in a database, accumulating experience on the current task to continuously refine the cost model for selecting the next promising *logical pipeline* and update the BO model for *physical pipeline* selection.

2.2 Evaluation Protocol Definition

One of the key challenges addressed in `VirnyFlow` is the definition of a comprehensive and flexible *evaluation protocol*, an aspect often overlooked by AutoML systems. As emphasized in Section 1, defining an *evaluation protocol* requires a deep understanding of the real-world problem and should be guided by human expertise. It must also support multiple dimensions of model performance, enable the definition of binary and intersectional groups, and incorporate a broad range of fairness metrics with possible extensibility.

To achieve this, our evaluation module is built on top of `Virny` [37], a Python library designed for in-depth model performance profiling across multiple dimensions, including accuracy, stability, uncertainty, and fairness. `Virny` is compatible with most tabular ML pipelines and provides ten fairness metrics[1], including widely used ones like *Equalized Odds* [36], as well as newer stability-based and uncertainty-based metrics such as *Label Stability Difference* [45].

[1]https://dataresponsibly.github.io/Virny/glossary/disparity_performance_dimensions/

In `VirnyFlow`, an *evaluation protocol* is defined as a part of an *experiment config*. Listing 1 illustrates how subgroups, metrics, and their respective weights for multi-objective optimization can be defined under `optimisation_args`, along with configurations for fairness and uncertainty quantification under `virny_args`.

2.3 Logical Pipeline Selection

After defining a flexible *evaluation protocol* and a broad *search space*, the next step is to use the available budget (e.g., time or number of pipeline executions, per *experiment config*) effectively by prioritizing the most promising *logical pipelines* and reducing effort on less promising ones. To achieve this goal, we integrate query optimization concepts from `Alpine Meadow` into `VirnyFlow`, re-implementing them from scratch and adapting them for multi-objective optimization. A key distinction between AutoML optimization and traditional query optimization, as highlighted by `Alpine Meadow` [65], is that "for ML pipelines we can actually try and evaluate hundreds if not thousands of pipelines, while in query optimization once a plan is executed there is nothing left to try out." Consequently, our optimizer *iteratively* selects and evaluates logical plans to maximize the likelihood of discovering a well-performing physical pipeline under multiple objectives.

Selection Strategy. Logical pipeline selection is of the optimizer is formulated as a three-step *Multi-Armed Bandit* problem: *(1 - Select)* an arm (i.e., *logical pipeline*) to run randomly but proportionally to the score. *(2 - Store)* execution history in the database. *(3 - Adjust)* scores, repeat from step (1). The *exploration factor* controls the probability of selecting the currently top-scoring *logical pipeline* or choosing an unfinished *logical pipeline* for execution, see Algorithm 2 in the Appendix of the full version of the paper [38].

Scoring Model. The score of a *logical pipeline* plan is defined as:

$$s = \sum_{i=1}^{n} w_i \cdot \mu_i + \frac{\theta}{c} \cdot \sum_{i=1}^{n} w_i \cdot \delta_i \quad (1)$$

where n is the number of optimization objectives, w_i is the weight assigned to each objective, μ_i and δ_i are the mean and standard deviation of the *logical pipeline* plan quality across multiple objectives, and c is the cost, or execution time, for a *logical pipeline* based on past history. The parameter θ acts as a *risk factor* that determines how much variance is tolerated when selecting a pipeline. A higher θ increases the likelihood of selecting pipelines with high variance. The variance term is normalized by execution time, ensuring that we are willing to wait longer for potentially higher rewards. However, the mean term is not adjusted by execution time, meaning that a pipeline with consistently strong performance should be prioritized early to ensure interactivity with the user. Section 2.5 details the interactivity and pipeline pruning logic in `VirnyFlow`.

This scoring model relies exclusively on historical execution data for the current dataset and task, and does not transfer of experience from other datasets. The challenge of meta-learning from other datasets and tasks in a multi-objective setting is significantly more complex than in the single-objective case studied in `Alpine Meadow`. We leave this as an interesting direction for future work.

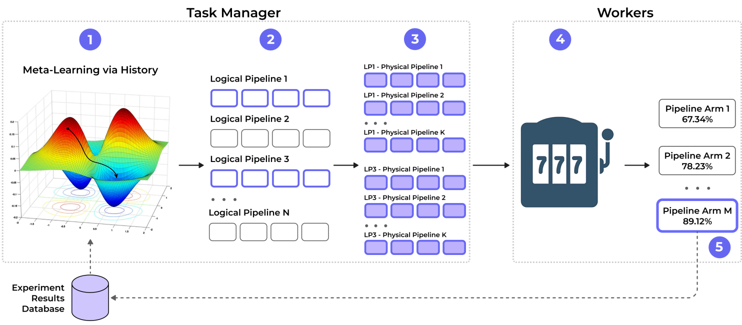


Figure 1: The optimization process: (1) search space definition, (2) logical pipeline selection, (3) physical pipeline selection, (4) pipeline evaluation and pruning, (5) search space model update.

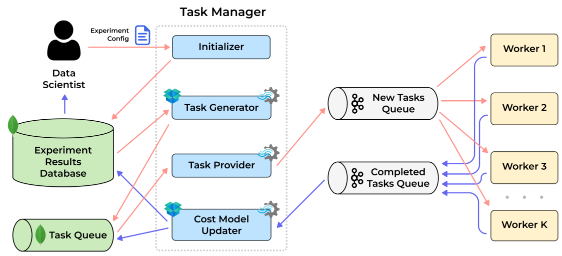


Figure 2: The `VirnyFlow` architecture. MongoDB components shown in green, Kafka queues in gray, workers in yellow, and `Task Manager` modules in blue. `Task Provider`, `Task Generator`, and `Cost Model Updater` use OpenBox for multi-stage, multi-objective optimization.

2.4 Physical Pipeline Selection

When the optimizer selects the next promising *logical pipeline*, it is instantiated into *k physical pipelines*, introducing random candidates to avoid getting stuck in local optima. A *physical pipeline* is a complete solution to the user-defined problem, represented as a DAG of primitives with fixed hyper-parameters. Each *physical pipeline* is generated using multi-objective Bayesian optimization (BO) [3, 30, 32] to tune the pipeline across multiple stages, including data cleaning and the use of fairness-enhancing interventions, and multiple objectives, including predictive accuracy, fairness, and stability. See Appendix A in the full version of the paper [38] for additional information on BO.

`VirnyFlow` uses `OpenBox` [47], a framework that offers a standardized set of single- and multi-objective BO optimizers (e.g., EI [49], EHVI [25], MESMO [10]), including support for constraints and parallelization, which aligns with `VirnyFlow`'s architecture.

When a user provides an *experiment config* (see Listing 1) and defines a search space of pipeline components and their hyper-parameters for each stage of the fixed-structure pipeline, `Task Manager` initializes all combinations of *logical pipelines* (see Step 2 in Figure 1) and assigns an individual *BO-advisor* to each for tuning, according to the optimization criterion. Each *BO-advisor* generates a *suggestion* of hyper-parameters for a *logical pipeline*, which is instantiated as a *physical pipeline* (Step 3). Importantly, each time the *BO-advisor* instantiates a *physical pipeline*, it jointly tunes multiple stages of the *logical pipeline* to align them with the optimization criterion, leveraging cross-stage interactions to improve overall performance (see motivation in Section 1.1). The *physical pipeline* is then evaluated on the *worker* side (Step 4), and the results are stored in the form of an *observation* in a database (Step 5). Based on these *observations*, the *BO-advisor* updates its model to generate the next promising *suggestion*. To incorporate user-defined weights into pipeline tuning, objectives are multiplied by the corresponding weights before being passed to the *BO-advisor*. Tuning continues until the maximum number of executed pipelines or the maximum time budget is reached.

Using the proposed optimization approach, `VirnyFlow` can simultaneously tune model fairness and stability together with predictive accuracy. For example, in the practical scenario described in Section 1, the system can optimize the *Selection Rate Difference (SRD)* [18, 42, 43], which quantifies the gap in selection rates between groups, and *Label Stability (LS)* [20, 45], which measures the disagreement among identical models (same type, architecture, and hyper-parameters) trained on bootstrap samples of the training data (see Section 3.1 for definitions).

Tuning fairness and stability metrics is challenging unless they are explicitly incorporated into the optimization problem throughout the *entire ML pipeline* (as is done in `VirnyFlow`), for several reasons [4, 34, 45, 63, 66, 77]. First, fairness constraints are non-differentiable and often conflict with accuracy [77], so optimizing for accuracy alone can move solutions away from the Bayes-optimal boundary. Second, limited data for minority groups [67, 77] allows optimizers to favor majority performance while neglecting minorities. Third, noisier data for minority groups increases uncertainty and harms model stability unless addressed during preprocessing [71]. Fourth, adding complex pipeline components, such as deep-learning-based preprocessors, can compound instability [4].

2.5 Pipeline Evaluation and Interactivity

Pipeline Evaluation. To enable incremental computation and early termination of unpromising pipelines, we adopt the *Adaptive Pipeline Selection* (APS) algorithm from `Alpine Meadow` [65],

Algorithm 1: Adaptive Pipeline Selection (APS)

Input: Pipeline $pipeline$, dataset $\mathcal{D}$.
Output: Score (negation of error), test objective metrics.

```
Split 𝒟 into 𝒟_train and 𝒟_test
Split 𝒟_train into equal-sized 𝒟^1_train, ..., 𝒟^N_train
foreach i ∈ 1...N do
    Train pipeline on 𝒟^{1...i}_train
    err_test, observation_test ← Test pipeline on 𝒟_test
    if err_test < err_best then
        err_best ← err_test
    yield err_test, observation_test
    err_train, observation_train ← Test pipeline on 𝒟^{1...i}_train
    if err_train > err_best then
        return err_test, observation_test
return err_test, observation_test
```

a bandit-based pruning strategy that detects poorly performing pipelines without utilizing the entire training set. We extend APS, presented in Algorithm 1, to support multiple objectives.

In lines 1–2, the algorithm splits the dataset $\mathcal{D}$ into training and test sets, followed by subsampling the training set in increments (e.g., starting at 50% and increasing by 10%). This reduces execution costs, similar to successive halving [39]. The halting criterion in APS is based on the idea that if the partial training error of a pipeline exceeds the best test error observed so far, the pipeline is terminated. We extend this principle to multiple objectives by computing err_{train} and err_{test} as a weighted sum of errors across different objectives, using user-defined weights (e.g., Listing 1). The variable $observation_{test}$ stores non-aggregated performance metrics of a pipeline, which are later used to update the score of the *logical pipeline* and stored in the database.

Interactivity. Interactivity is embedded into both the scoring model and the pipeline pruning logic, ensuring that promising results are presented to users earlier. Additionally, `VirnyFlow` integrates visualization capabilities from `Virny` and `OpenBox`, allowing users to track experiment progress and pipeline tuning across multiple performance dimensions (see Figures 7–9 in Appendix D of [38]). This helps users assess the correctness of their *experiment config* early and make necessary adjustments. Enhancing `VirnyFlow` with more advanced visualization interfaces to better support users in selecting fairness metrics, constraints, and optimizers for real-world applications is an interesting avenue for future work.

2.6 Distributed Execution

To enhance scalability and efficiency, `VirnyFlow` combines distributed execution, fine-grained parallelism, asynchronous communication, and asynchronous programming. Figure 2 presents the architecture of `VirnyFlow`, which is implemented in Python and consists of `Task Manager`, `Workers`, `Distributed Queue`, and an external database (MongoDB [50]). Within `Task Manager`, there are four key components: `Initializer`, `Task Generator`, `Task Provider`, and `Cost Model Updater`, with the last three functioning as asynchronous data processors.

The execution process begins when a user provides an *experiment config* and a search space for tuning to `Task Manager`. The `Initializer` then instantiates all *logical pipelines* in the external database. Next, the `Task Generator` initializes an individual *BO-advisor* for each *logical pipeline*, applies a cost model to select a promising *logical pipeline* (see Section 2.3 for details), and uses the *BO-advisor* to generate an optimal set of hyper-parameters, forming a *physical pipeline* (see Section 2.4 for details). This *physical pipeline* is then packaged into a *task* and stored in the `Task Queue` in an external database.

Similar to `Alpine Meadow` [65], our `Task Queue` has a limited size of m, meaning that tasks are continuously added by the `Task Generator` until this limit is reached. This mechanism ensures that workers always have enough preloaded tasks available once they complete their current assignments. Additionally, as discussed in Section 2.1, the *BO-advisor* generates not just one *suggestion* but k *suggestions*, introducing random candidates to mitigate the risk of getting stuck in a local optimum. Thus, when k slots become available in the queue, the next promising *logical pipeline* is selected, and k *physical pipelines* are added to the queue. According to `Alpine Meadow`, this approach is effective under the assumption that the number of workers w is significantly larger than k, i.e., $w \gg k$, and that the queue size m is greater than w.

The reason for first storing tasks in an external database rather than sending them directly to the `Distributed Queue` is fault tolerance. If a system failure occurs and any component shuts down, the execution progress and results remain intact in the database. In such cases, a user can restart `VirnyFlow` and resume execution from the last saved state. To add tasks from the `Task Queue` to the `Distributed Queue`, `Task Manager` uses the `Task Provider`.

To enable efficient communication between `Task Manager` and `Workers`, `VirnyFlow` employs `Distributed Queue` built on Apache Kafka [2], a distributed event streaming platform. Asynchronous communication is achieved using two queues: the `New Tasks Queue`, which contains new tasks for workers to execute, and the `Completed Tasks Queue`, which stores execution results (*observations*). Note that the combination of fine-grained parallelism, where pipelines are executed as independent *tasks*, and asynchronous communication via `Distributed Queue` ensures efficient resource utilization. This is because the computational cost of executing a single *task* is relatively low, and each `Worker` retrieves the next available task from the queue as soon as it completes the previous one. Next, execution results are processed by the `Cost Model Updater`, which reads from the `Completed Tasks Queue` and updates both the global cost model and the corresponding *BO-advisor* for the associated *logical pipeline*. All execution progress and results are stored in the database.

`VirnyFlow` incorporates additional optimizations. `Task Manager` components are implemented using asynchronous programming with `asyncio`[2], ensuring non-blocking execution. The `Cost Model Updater` uses just-in-time (JIT) compilation via `numba`[3] to rapidly recompute *logical pipeline* scores based on new *observations*. To further optimize performance, `VirnyFlow` minimizes the number of queries to the external database and indexes all database tables to accelerate retrieval.

3 EXPERIMENTS

Our experiments aim to answer the following research questions:

- **RQ1** Is `VirnyFlow` able to optimize ML pipelines according to different multi-objective optimization criteria (Section 3.2)?
- **RQ2** How does `VirnyFlow` compare to state-of-the-art AutoML systems in terms of performance (**RQ2.1**, Section 3.3) and scalability (**RQ2.2**, Section 3.4)?
- **RQ3** What is the sensitivity of `VirnyFlow` to different configuration settings (Section 3.5)?

3.1 Experimental Setup

Datasets. We conduct experiments on five datasets from diverse social decision-making contexts, including hiring, healthcare, and public insurance coverage, summarized in Table 1. Each dataset is associated with a binary classification task, where a positive label represents access to a desirable social good (e.g., employment, insurance, or healthcare). We selected these datasets to ensure

[2]https://docs.python.org/3.13/library/asyncio.html
[3]https://numba.pydata.org/

Table 1: Dataset information.

name	domain	# tuples	# attrs	sensitive attrs
`diabetes`	healthcare	952	17	sex
`folk-emp`	hiring	15,000	16	sex, race
`folk-pubcov`	public coverage	50,000	19	sex, race
`heart`	healthcare	70,000	11	sex
`folk-emp-big`	hiring	200,000	16	sex, race

broad coverage of social domains, dataset sizes, and optimization objectives. Datasets are randomly split into 80% training and 20% test if a dataset size is greater than 1,000 rows, otherwise, a 70%:30% ratio is used. Dataset are summarized in Table 1, with detailed descriptions deferred to Appendix B.1 of the full paper [38].

Baselines. We compare our system with two state-of-the-art, broad-spectrum AutoML baselines: `Alpine Meadow` [65] and `auto-sklearn` [29], both highlighted in recent benchmarks [33, 52] and surveys [7, 8] (see Appendix B.4 in [38]). For a fair comparison, we use the system configurations as specified in their original papers and standardize the search space across all systems. This includes a common set of ML models (`dt_clf`, `lr_clf`, `rf_clf`, `xgb_clf`, `lgbm_clf`) and their hyperparameters (see Appendix B.3 in [38]). We do not compare with recent fairness-aware AutoML systems [21, 24, 46, 53, 59, 62, 78] because they either lack publicly available code, do not support our required fairness and stability metrics in the optimization process, do not handle intersectional groups, or do not consider scalability in their architectures.

Metrics for Comparison. Which optimization criteria are more relevant to a given problem depends on the application domain and the stakeholders involved [37, 51, 79]. Therefore, for each dataset, we select the most appropriate set of metrics from those listed below, covering various performance dimensions defined in the *evaluation protocol* for `VirnyFlow`. In all experiments, our system uses EHVI [25] as an MOBO optimizer to jointly tune this set of metrics. In contrast, other systems optimize only for the *F1 score*.

To assess accuracy, we report the *F1 score* because it is a more reliable metric than accuracy for imbalanced data.

To assess stability, we present the average *Label Stability (LS)* [20, 45] across the entire test set. For binary classification, this is calculated for each sample using *Label Stability* = $\frac{|B_+ - B_-|}{B}$, where B_+ represents the frequency with which the sample is classified as positive, B_- indicates how often it is classified as negative, and $B = B_+ + B_-$ results from models trained on bootstrapped samples of the training set. In all our experiments, we use $B = 50$ and set the bootstrap fraction to 80%.

To assess fairness, we report error disparity metrics based on group-specific error rates, namely *True Positive Rate Difference (TPRD)*, *True Negative Rate Difference (TNRD)*, *False Negative Rate Difference (FNRD)*, and *Selection Rate Difference (SRD)*, see Appendix B.2 of the full paper [38] for definitions.

Lastly, we evaluate efficiency using two metrics: *runtime* (in seconds) and *speedup*. *Runtime* refers to the total time a system takes to evaluate a fixed number of pipelines and return the final ML pipeline. *Speedup* is defined as the ratio between the runtime of `VirnyFlow` using a single worker/CPU and the runtime of the system using k workers/CPUs. For consistency, when computing *speedup* for `Alpine Meadow` and `auto-sklearn`, we also use `VirnyFlow`'s single-worker runtime as the baseline to ensure a uniform basis for comparison. Details on the computing infrastructure are available in Appendix B.5 of the full paper [38].

3.2 Functionality of `VirnyFlow`

In this section, we evaluate the functionality of `VirnyFlow`, demonstrating its ability to optimize ML pipelines based on multiple objectives (**RQ1**). We conduct case studies on three datasets of varying sizes and domains: `diabetes`, `folk-emp`, and `folk-pubcov`. The search space includes four models (`lr_clf`, `rf_clf`, `lgbm_clf`, and `gandalf_clf`), along with their respective hyperparameter grids (details in Appendix B.3 in [38]), and does not include feature engineering and fairness-enhancing interventions in this evaluation.

To highlight `VirnyFlow` 's capability for multi-objective optimization, including fairness and stability, we define different metric

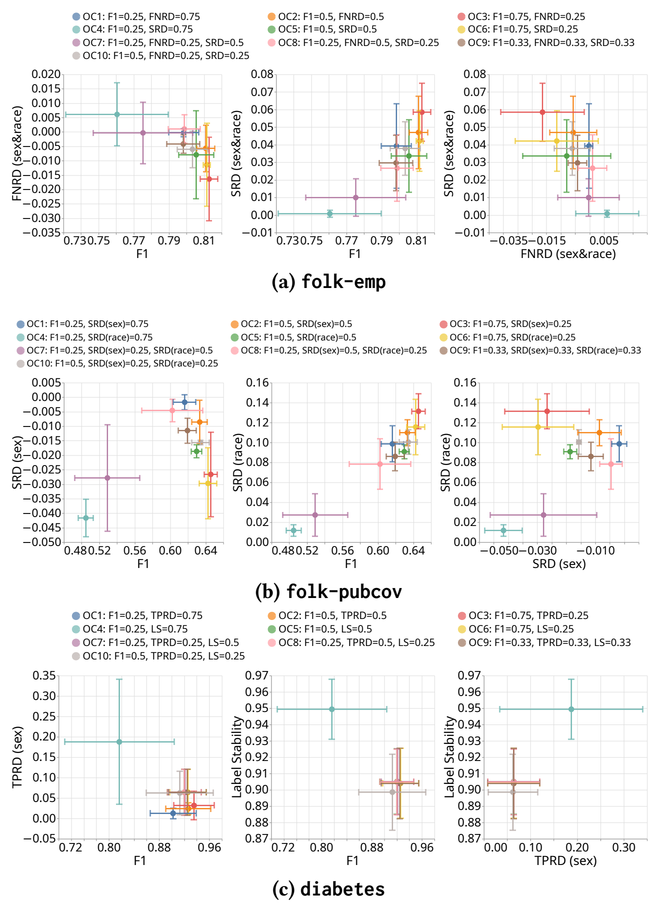


Figure 3: Mean and standard deviation of performance metrics from multi-objective tuning for (a) `folk-emp`, (b) `folk-pubcov`, and (c) `diabetes`. Each point represents an optimization criterion (OC), as labeled in the legend. For TPRD, FNRD, and SRD, values closer to zero indicate better fairness.

Table 2: Mean and standard deviation of the performance metrics for the best pipeline identified by each system. Bold text marks the best value per metric, while gray shading shows the best *average score* across multiple metrics using equal weighting.

System	**Diabetes (≈1K)**		**Folk Employment (15K)**		**Folk Public Coverage (50K)**		
	F1	Label Stability	F1	FNRD (sex & race)	F1	SRD (sex)	SRD (race)
autosklearn	0.892 ±0.049	0.904 ±0.017	0.808 ±0.004	0.012 ±0.019	**0.633 ±0.009**	-0.033 ±0.010	0.177 ±0.008
alpine_meadow	0.915 ±0.034	0.889 ±0.011	0.809 ±0.007	0.014 ±0.023	0.628 ±0.006	-0.033 ±0.008	0.178 ±0.011
virny_flow	**0.929 ±0.023**	**0.908 ±0.019**	**0.811 ±0.005**	**-0.010 ±0.010**	0.596 ±0.029	**-0.011 ±0.007**	**0.068 ±0.029**

sets tailored to each dataset. To assess how VirnyFlow handles prioritization among objectives, we test different weight combinations (summing to 1) for each set of metrics. Each dataset is evaluated using 10 optimization criteria, shown in the legend of Figure 3. Optimization runs for up to 800 trials (i.e., candidate pipelines) per dataset and seed, and the best pipeline produced by VirnyFlow is used for evaluation. The number of suggestions k per *logical pipeline* is set to 2, and training set fractions for halting are set to {0.7, 1.0} (both parameters are described in Section 2.6).

Figure 3 shows the mean and standard deviation of the best-performing ML pipelines optimized using different criteria across datasets. For fairness metrics TPRD, FNRD, and SRD, values closer to zero indicate better fairness. Each metric is reported regardless of whether it was included in the optimization objective, allowing comparison across settings.

Figure 3a shows that VirnyFlow effectively optimizes both fairness and accuracy, even for intersectional groups (e.g., sex & race). In Figure 3a-left, optimization criterion OC1 (favoring FNRD over F1) achieves perfect fairness (FNRD = 0.0) with only a small accuracy drop (F1 decreases by 0.012). Similarly, Figure 3a-center shows that OC4 (prioritizing SRD over F1) also achieves SRD = 0.0 but at a greater F1 cost (drop of 0.05), suggesting SRD is harder to optimize on folk-emp. Importantly, VirnyFlow can jointly optimize all three metrics: OC9 (equal weights) yields F1 = 0.80, FNRD = -0.004, and SRD = 0.03, each close to the respective best, and demonstrates strong trade-off performance suitable for production use.

Figure 3b shows optimization results on folk-pubcov, where SRD is optimized separately for sex and race alongside F1. The plots show that VirnyFlow can optimize all three metrics, but trade-offs arise. As shown in Figure 3b-center, jointly optimizing SRD for both groups (OC7) is more difficult and leads to a notable F1 drop (≈0.11). In contrast, OC9 incurs only a minor F1 loss (≈0.02) but yields significantly worse SRD for race (by 0.06) compared to OC7. This illustrates a clear trade-off between fairness and accuracy—specifically, between SRD for race and F1. Ultimately, which objective to prioritize depends on the problem's context and constraints.

Finally, Figure 3c shows optimization results for the diabetes dataset, where F1, LS, and TPRD for sex are optimized using different weightings. The plots demonstrate that VirnyFlow can effectively optimize LS; for example, OC4 (which prioritizes stability) achieves an LS value 0.045 better than any other configuration (Figure 3c-center). To our knowledge, VirnyFlow is the first ML system capable of tuning for model stability. While this gain comes with trade-offs, such as lower F1 and TPRD, VirnyFlow makes these trade-offs transparent. Moreover, built on top of OpenBox and equipped with a standardized API for model tuners, VirnyFlow is easily extensible, serving as a flexible playground for advancing model stability without sacrificing other performance dimensions.

In summary, we answer **RQ1** affirmatively: VirnyFlow effectively optimizes ML pipelines under diverse multi-objective criteria, including fairness across binary and intersectional groups and model stability. This demonstrates its potential to support context-sensitive pipeline development.

3.3 Performance of VirnyFlow vs. Other Systems

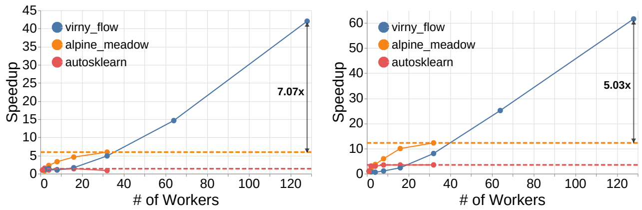
In this experiment, we compare the performance of our system with state-of-the-art AutoML systems (**RQ2.1**). Specifically, we evaluate VirnyFlow against Alpine Meadow and auto-sklearn on three datasets of varying sizes: diabetes, folk-emp, and folk-pubcov. Each dataset is associated with its own optimization criterion. All systems operate within the same search space and are allocated identical computational resources, as described in Section 3.1. Each system is run for 60 minutes per dataset and seed to discover the best-performing ML pipeline. Here, VirnyFlow performs multi-objective optimization using equal weights for each selected metric, while Alpine Meadow and auto-sklearn optimize solely for F1.

Table 2 reports the mean and standard deviation of the performance metrics for the best pipeline identified by each system. Bold text indicates the best value for individual metrics, while gray shading highlights the best *average score* across multiple metrics using equal weighting. The results show that VirnyFlow consistently outperforms both Alpine Meadow and auto-sklearn in terms of the *average score* across all three datasets. On diabetes and folk-emp, VirnyFlow clearly outperforms the baselines across all metrics. For folk-pubcov, VirnyFlow achieves a slightly lower F1 score (by 0.037) but significantly improves fairness metrics, particularly SRD for race (by 0.11), demonstrating a favorable trade-off.

In summary, in response to **RQ2.1** (performance), we find that VirnyFlow outperforms both Alpine Meadow and auto-sklearn.

3.4 Scalability of VirnyFlow vs. Other Systems

In this section, we evaluate the scalability of VirnyFlow compared to state-of-the-art AutoML systems (**RQ2.2**). We assess how each system scales with increasing computational resources by varying the number of workers (one CPU core per worker) and measuring the resulting *speedup* (Section 3.1) on two large datasets: heart (70K samples) and folk-emp-big (200K samples), each with a distinct optimization criterion. The number of workers increases exponentially across configurations. Since Alpine Meadow and auto-sklearn are limited to single-node execution, their evaluations cap at 32 workers (i.e., one machine). In contrast, VirnyFlow supports distributed parallelism and is tested with up to 128 workers across four nodes



(a) `heart` (70K) **(b) `folk-emp-big` (200K)**

Figure 4: Scalability study with varying numbers of workers (one CPU core per worker). `Alpine Meadow` and `auto-sklearn` do not scale beyond 32 workers due to their architectural limitations that prevent multi-node execution.

to simulate a realistic multi-node setup. For fairness, each system executes 200 pipelines per configuration. All other experimental settings follow Section 3.3.

Figure 4 shows the *speedup* achieved by each system, with dashed lines marking the best speedup per configuration. On the `heart` dataset, `auto-sklearn` achieves peak speedup with 8 workers. Overall, `VirnyFlow` consistently outperforms both `Alpine Meadow` and `auto-sklearn` in runtime, reaching speedups of up to 5.03 on `folk-emp-big` and 7.07 on `heart` with 128 workers, substantially surpassing `Alpine Meadow` at its 32-worker maximum. Even when limited to 32 workers, `VirnyFlow` outperforms `auto-sklearn` and remains competitive with `Alpine Meadow`, despite lacking `Alpine Meadow` 's low-level code optimizations. `VirnyFlow` is less efficient with fewer than 8 workers due to resource overhead from Apache Kafka [2]. All systems were allocated equal total resources. Improving low-level efficiency for small-scale deployments is a promising direction for future work.

In summary, in response to **RQ2.2** (scalability), we find that `VirnyFlow` outperforms both `Alpine Meadow` and `auto-sklearn`, particularly at scale with many workers and distributed execution. Figures 5 and 6 in Appendix C.1 in [38] show accuracy and fairness results, confirming that `VirnyFlow` achieves higher *average scores* across multiple objectives on large datasets, consistent with its performance on smaller datasets (Section 3.3).

Table 3: Sensitivity of `VirnyFlow` to the number of physical pipeline candidates (k) per logical pipeline selection. Gray shadowing highlights the most optimal setting in terms of performance and efficiency.

# of PPs	Heart (70K)		Folk Pub. Cov. (50K)	
	Score	Runtime	Score	Runtime
1	86.31 ±0.20	1049 ±461	83.05 ±0.29	1411 ±587
2	86.30 ±0.21	917 ±180	82.90 ±0.30	1095 ±315
4	86.28 ±0.33	781 ±104	82.97 ±0.34	1036 ±284
8	86.17 ±0.14	786 ±276	83.02 ±0.22	1261 ±430
16	86.08 ±0.61	724 ±175	82.82 ±0.35	909 ±146
32	85.95 ±0.91	585 ±165	82.91 ±0.48	868 ±409

3.5 System Configuration Sensitivity

Finally, we examine the sensitivity of `VirnyFlow` to different configuration settings (**RQ3**). Table 3 reports results for varying the number of physical pipeline candidates (k) per logical pipeline selection (Section 2.6), which affects execution parallelism and the exploration–exploitation trade-off. Table 9 presents results for different training set fractions used in pipeline pruning. Experiments are conducted on two datasets using a single optimization criterion with equal weights for all component metrics. For the `heart` dataset, the objective averages F1 and TNRD; for `folk-pubcov`, it averages F1, SRD for sex, and SRD for race. The search space matches that of Sections 3.3 and 3.4, with two added fairness-enhancing interventions: Disparate Impact Remover [28] and Adversarial Debiasing [81]. Each configuration evaluates up to 200 pipelines.

Each table reports the mean and standard deviation of the composite objective score, as well as runtime per system run. Table 3 shows a clear trade-off: increasing the number of physical pipeline candidates (k) reduces runtime but generally degrades performance. Notably, $k = 4$ offers a strong balance between efficiency and effectiveness across both datasets. Table 9 indicates that using multiple training set fractions for pruning does not consistently reduce runtime and may even increase it. Additionally, very small pruning fractions can hurt performance. For example, the {0.5, 1.0} setting yields the best results on `heart`, while {0.75, 1.0} achieves a better trade-off on the smaller `folk-pubcov` dataset.

In summary, in response to **RQ3** (sensitivity to configuration), we show the trade-off between pipeline performance and runtime, and propose default settings as a strong baseline for further tuning.

4 CONCLUSION AND FUTURE WORK

Conclusion. This paper introduces `VirnyFlow`, the first *design space* for responsible model development, aimed at helping data scientists build ML pipelines that are customized to the specific context of their problems. By integrating a context-sensitive *evaluation protocol*, we enable multi-stage, multi-objective optimization that goes beyond traditional performance metrics. Our approach unifies diverse techniques, including multi-objective Bayesian optimization, cost-aware multi-armed bandits, query optimization, and distributed parallelism, into an interactive, flexible, and efficient *design space* for experimentation. Extensive empirical evaluation demonstrates that `VirnyFlow` achieves superior performance and scalability compared to existing state-of-the-art AutoML solutions, introducing a novel perspective on responsible ML systems and establishing a strong foundation for future research.

Future Work. Several directions remain open for future work. First, we plan to explore advanced scoring methods for multi-objective optimization. While `VirnyFlow` currently uses a simple weighted sum, alternatives such as Tchebycheff, ϵ-constraint, and model-based techniques [44] may offer performance gains. Second, growing interest in incorporating decision-maker preferences [15, 75] motivates future enhancements to `VirnyFlow` 's visualization interfaces to better support metric selection, constraint specification, and optimizer configuration. Finally, although `VirnyFlow` supports multi-node execution and scales well, we aim to refine pruning mechanisms for multi-objective optimization and further improve resource efficiency.

REFERENCES

[1] Meta AI. 2025. Ax: Adaptive Experimentation Platform. https://ax.dev. Accessed: 2025-03-01.

[2] Apache Software Foundation. 2024. *Apache Kafka: A Distributed Event Streaming Platform.* https://kafka.apache.org/ Version 3.6.

[3] Francesco Archetti and Antonio Candelieri. 2019. *Bayesian optimization and data science.* Vol. 849. Springer.

[4] Falaah Arif Khan, Denys Herasymuk, Nazar Protsiv, and Julia Stoyanovich. 2025. Still More Shades of Null: An Evaluation Suite for Responsible Missing Value Imputation. *Proc. VLDB Endow.* (2025). https://arxiv.org/pdf/2409.07510 accepted for publication, to appear.

[5] Falaah Arif Khan, Denys Herasymuk, and Julia Stoyanovich. 2023. On Fairness and Stability: Is Estimator Variance a Friend or a Foe? *CoRR* abs/2302.04525 (2023). https://doi.org/10.48550/ARXIV.2302.04525 arXiv:2302.04525

[6] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. 2020. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in neural information processing systems* 33 (2020), 21524–21538.

[7] Mitra Baratchi, Can Wang, Steffen Limmer, Jan N van Rijn, Holger Hoos, Thomas Bäck, and Markus Olhofer. 2024. Automated machine learning: past, present and future. *Artificial Intelligence Review* 57, 5 (2024), 1–88.

[8] Rafael Barbudo, Sebastián Ventura, and José Raúl Romero. 2023. Eight years of AutoML: categorisation, review and trends. *Knowledge and Information Systems* 65, 12 (2023), 5097–5149.

[9] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2023. *Fairness and machine learning: Limitations and opportunities.* MIT press.

[10] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. 2019. Max-value entropy search for multi-objective Bayesian optimization. *Advances in neural information processing systems* 32 (2019).

[11] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. 2019. Max-value Entropy Search for Multi-Objective Bayesian Optimization. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/82edc5c9e21035674d481640448049f3-Paper.pdf

[12] Syrine Belakaria, Aryan Deshwal, Nitthilan Kannappan Jayakodi, and Janardhan Rao Doppa. 2020. Uncertainty-Aware Search Framework for Multi-Objective Bayesian Optimization. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 06 (April 2020), 10044–10052. https://doi.org/10.1609/aaai.v34i06.6561

[13] Emily Black, Klas Leino, and Matt Fredrikson. 2021. Selective ensembles for consistent predictions. *arXiv preprint arXiv:2111.08230* (2021).

[14] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems* 29 (2016).

[15] Juergen Branke. 2016. *MCDA and Multiobjective Evolutionary Algorithms.* Springer New York, New York, NY, 977–1008. https://doi.org/10.1007/978-1-4939-3094-4_23

[16] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, et al. 2013. API design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238* (2013).

[17] Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency.* PMLR, 77–91.

[18] Toon Calders and Sicco Verwer. 2010. Three naive bayes approaches for discrimination-free classification. *Data mining and knowledge discovery* 21 (2010), 277–292.

[19] Antonio Candelieri, Andrea Ponti, and Francesco Archetti. 2024. Fair and green hyperparameter optimization via multi-objective and multiple information source Bayesian optimization. *Machine Learning* 113, 5 (2024), 2701–2731.

[20] Michael C. Darling and David J. Stracuzzi. 2018. Toward Uncertainty Quantification for Supervised Classification.

[21] Richeek Das and Samuel Dooley. 2023. Fairer and more accurate tabular models through nas. *arXiv preprint arXiv:2310.12145* (2023).

[22] Jeffrey Dastin. 2022. Amazon scraps secret AI recruiting tool that showed bias against women. In *Ethics of data and analytics.* Auerbach Publications, 296–299.

[23] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. 2021. Retiring adult: New datasets for fair machine learning. *NeurIPS* 34 (2021), 6478–6490.

[24] Samuel Dooley, Rhea Sukthanker, John Dickerson, Colin White, Frank Hutter, and Micah Goldblum. 2024. Rethinking bias mitigation: Fairer architectures make for fairer face recognition. *Advances in Neural Information Processing Systems* 36 (2024).

[25] Michael TM Emmerich, Kyriakos C Giannakoglou, and Boris Naujoks. 2006. Single-and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation* 10, 4 (2006), 421–439.

[26] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505* (2020).

[27] A Feder Cooper, Solon Barocas, Christopher De Sa, and Siddhartha Sen. 2023. Variance, Self-Consistency, and Arbitrariness in Fair Classification. *arXiv e-prints* (2023), arXiv–2301.

[28] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining.* 259–268.

[29] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2022. Auto-sklearn 2.0: Hands-free automl via meta-learning. *Journal of Machine Learning Research* 23, 261 (2022), 1–61.

[30] Peter I Frazier. 2018. Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems.* Informs, 255–278.

[31] Yarin Gal et al. 2016. Uncertainty in deep learning. (2016).

[32] Roman Garnett. 2023. *Bayesian optimization.* Cambridge University Press.

[33] Pieter Gijsbers, Marcos LP Bueno, Stefan Coors, Erin LeDell, Sébastien Poirier, Janek Thomas, Bernd Bischl, and Joaquin Vanschoren. 2024. Amlb: an automl benchmark. *Journal of Machine Learning Research* 25, 101 (2024), 1–65.

[34] Shubha Guha, Falaah Arif Khan, Julia Stoyanovich, and Sebastian Schelter. 2024. Automated data cleaning can hurt fairness in machine learning-based decision making. *IEEE Transactions on Knowledge and Data Engineering* 36, 12 (2024), 7368–7379.

[35] Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michèle Sebag, et al. 2019. Analysis of the automl challenge series. *Automated Machine Learning* 177 (2019), 177–219.

[36] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. *Advances in neural information processing systems* 29 (2016).

[37] Denys Herasymuk, Falaah Arif Khan, and Julia Stoyanovich. 2024. Responsible Model Selection with Virny and VirnyView. In *Companion of the 2024 International Conference on Management of Data.* 488–491.

[38] Denys Herasymuk, Nazar Protsiv, and Julia Stoyanovich. 2025. VirnyFlow: A Design Space for Responsible Model Development [full version]. https://update. Accessed: 2025-05-30.

[39] Kevin Jamieson and Ameet Talwalkar. 2016. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial intelligence and statistics.* PMLR, 240–248.

[40] Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. *Journal of Global Optimization* 13, 4 (1998), 455–492. https://doi.org/10.1023/a:1008306431147

[41] Manu Joseph and Harsh Raj. 2022. GANDALF: gated adaptive network for deep automated learning of features. *arXiv preprint arXiv:2207.08548* (2022).

[42] Faisal Kamiran and Toon Calders. 2009. Classifying without discriminating. In *2009 2nd international conference on computer, control and communication.* IEEE, 1–6.

[43] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. 2011. Fairness-aware learning through regularization approach. In *2011 IEEE 11th international conference on data mining workshops.* IEEE, 643–650.

[44] Florian Karl, Tobias Pielok, Julia Moosbauer, Florian Pfisterer, Stefan Coors, Martin Binder, Lennart Schneider, Janek Thomas, Jakob Richter, Michel Lang, et al. 2023. Multi-objective hyperparameter optimization in machine learning—An overview. *ACM Transactions on Evolutionary Learning and Optimization* 3, 4 (2023), 1–50.

[45] Falaah Arif Khan, Denys Herasymuk, and Julia Stoyanovich. 2023. On Fairness and Stability: Is Estimator Variance a Friend or a Foe? *arXiv preprint arXiv:2302.04525* (2023).

[46] CR Komala, Ashok Kumar, N Hema, S Nagarani, Ajay Singh Yadav, M Rajendiran, R Srinivasan, and V Vijayan. 2024. Fair-AutoML: Enhancing fairness in machine learning predictions through automated machine learning and bias mitigation techniques. In *AIP Conference Proceedings*, Vol. 3193. AIP Publishing.

[47] Yang Li, Yu Shen, Wentao Zhang, Yuanwei Chen, Huaijun Jiang, Mingchao Liu, Jiawei Jiang, Jinyang Gao, Wentao Wu, Zhi Yang, et al. 2021. Openbox: A generalized black-box optimization service. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining.* 3209–3219.

[48] Yan Li, Matthew Sperrin, Darren M Ashcroft, and Tjeerd Pieter Van Staa. 2020. Consistency of variety of machine learning and statistical models in predicting clinical risks of individual patients: longitudinal cohort study using cardiovascular disease as exemplar. *BMJ* 371 (2020).

[49] Jonas Močkus. 1975. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974 6.* Springer, 400–404.

[50] MongoDB, Inc. 2024. *MongoDB: The Developer Data Platform.* https://www.mongodb.com/ Version 7.0.

[51] Arvind Narayanan. 2018. 21 Fairness Definitions and Their Politics. Available at https://www.cs.princeton.edu/~arvindn/talks.

[52] Felix Neutatz, Marius Lindauer, and Ziawasch Abedjan. 2024. AutoML in heavily constrained applications. *The VLDB Journal* 33, 4 (2024), 957–979.

[53] Giang Nguyen, Sumon Biswas, and Hridesh Rajan. 2023. Fix fairness, don't ruin accuracy: Performance aware fairness repair using automl. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 502–514.
[54] Nikolay O Nikitin, Pavel Vychuzhanin, Mikhail Sarafanov, Iana S Polonskaia, Ilia Revin, Irina V Barabanova, Gleb Maximov, Anna V Kalyuzhnaya, and Alexander Boukhanovsky. 2022. Automated evolutionary approach for the design of composite machine learning pipelines. *Future Generation Computer Systems* 127 (2022), 109–125.
[55] Ziad Obermeyer, Brian Powers, Christine Vogeli, and Sendhil Mullainathan. 2019. Dissecting racial bias in an algorithm used to manage the health of populations. *Science* 366, 6464 (2019), 447–453.
[56] Randal S Olson, Nathan Bartley, Ryan J Urbanowicz, and Jason H Moore. 2016. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the genetic and evolutionary computation conference 2016*. 485–492.
[57] Biswajit Paria, Kirthevasan Kandasamy, and Barnabás Póczos. 2020. A Flexible Framework for Multi-Objective Bayesian Optimization using Random Scalarizations. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference (Proceedings of Machine Learning Research)*, Ryan P. Adams and Vibhav Gogate (Eds.), Vol. 115. PMLR, 766–776. https://proceedings.mlr.press/v115/paria20a.html
[58] Valerio Perrone, Michele Donini, Muhammad Bilal Zafar, Robin Schmucker, Krishnaram Kenthapadi, and Cédric Archambeau. 2021. Fair bayesian optimization. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 854–863.
[59] Florian Pfisterer, Stefan Coors, Janek Thomas, and Bernd Bischl. 2019. Multi-objective automatic machine learning with autoxgboostmc. *arXiv preprint arXiv:1908.10796* (2019).
[60] Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and Bernd Bischl. 2022. Yahpo gym-an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In *International Conference on Automated Machine Learning*. PMLR, 3–1.
[61] Herilalaina Rakotoarison, Marc Schoenauer, and Michèle Sebag. 2019. Automated machine learning with monte-carlo tree search. *arXiv preprint arXiv:1906.00170* (2019).
[62] Jake Robertson, Thorsten Schmidt, Frank Hutter, and Noor Awad. 2024. A Human-in-the-Loop Fairness-Aware Model Selection Framework for Complex Fairness Objective Landscapes. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, Vol. 7. 1231–1242.
[63] Sebastian Schelter and Julia Stoyanovich. 2020. Taming Technical Bias in Machine Learning Pipelines. *IEEE Data Eng. Bull.* 43, 4 (2020). http://sites.computer.org/debull/A20dec/p39.pdf
[64] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* 104, 1 (2016), 148–175. https://doi.org/10.1109/JPROC.2015.2494218
[65] Zeyuan Shang, Emanuel Zgraggen, Benedetto Buratti, Ferdinand Kossmann, Philipp Eichmann, Yeounoh Chung, Carsten Binnig, Eli Upfal, and Tim Kraska. 2019. Democratizing data science through interactive curation of ml pipelines. In *Proceedings of the 2019 international conference on management of data*. 1171–1188.
[66] Julia Stoyanovich, Serge Abiteboul, Bill Howe, H. V. Jagadish, and Sebastian Schelter. 2022. Responsible data management. *Commun. ACM* 65, 6 (2022), 64–74. https://doi.org/10.1145/3488717
[67] Harini Suresh and John Guttag. 2021. A framework for understanding sources of harm throughout the machine learning life cycle. In *Proceedings of the 1st ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization*. 1–9.
[68] Shinya Suzuki, Shion Takeno, Tomoyuki Tamura, Kazuki Shitara, and Masayuki Karasuyama. 2020. Multi-objective Bayesian Optimization using Pareto-frontier Entropy. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Hal Daumé III and Aarti Singh (Eds.), Vol. 119. PMLR, 9279–9288. https://proceedings.mlr.press/v119/suzuki20a.html
[69] Thomas Swearingen, Will Drevo, Bennett Cyphers, Alfredo Cuesta-Infante, Arun Ross, and Kalyan Veeramachaneni. 2017. ATM: A distributed, collaborative, scalable system for automated machine learning. In *2017 IEEE international conference on big data (big data)*. IEEE, 151–162.
[70] Latanya Sweeney. 2013. Discrimination in online ad delivery. *Commun. ACM* 56, 5 (2013), 44–54.
[71] Anique Tahir, Lu Cheng, and Huan Liu. 2023. Fairness through aleatoric uncertainty. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2372–2381.
[72] Rachael Tatman. 2017. Gender and dialect bias in YouTube's automatic captions. In *Proceedings of the first ACL workshop on ethics in natural language processing*. 53–59.
[73] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 847–855.
[74] Neha Prerna Tigga and Shruti Garg. 2020. Prediction of Type 2 Diabetes using Machine Learning Classification Methods. *Procedia Computer Science* 167 (2020), 706–716. https://doi.org/10.1016/j.procs.2020.03.336 International Conference on Computational Intelligence and Data Science.
[75] Handing Wang, Markus Olhofer, and Yaochu Jin. 2017. A mini-review on preference modeling and articulation in multi-objective optimization: Current status and challenges. *Complex &; Intelligent Systems* 3, 4 (Aug 2017), 233–245. https://doi.org/10.1007/s40747-017-0053-9
[76] Hilde Weerts, Florian Pfisterer, Matthias Feurer, Katharina Eggensperger, Edward Bergman, Noor Awad, Joaquin Vanschoren, Mykola Pechenizkiy, Bernd Bischl, and Frank Hutter. 2024. Can fairness be automated? Guidelines and opportunities for fairness-aware AutoML. *Journal of Artificial Intelligence Research* 79 (2024), 639–677.
[77] Steven Euijong Whang, Yuji Roh, Hwanjun Song, and Jae-Gil Lee. 2023. Data collection and quality challenges in deep learning: A data-centric ai perspective. *The VLDB Journal* 32, 4 (2023), 791–813.
[78] Qingyun Wu and Chi Wang. 2021. FairAutoML: Embracing unfairness mitigation in AutoML. *arXiv preprint arXiv:2111.06495* (2021).
[79] Jie Xu, Yunyu Xiao, Wendy Hui Wang, Yue Ning, Elizabeth A Shenkman, Jiang Bian, and Fei Wang. 2022. Algorithmic fairness in computational medicine. *EBioMedicine* 84 (2022).
[80] Anatoly Yakovlev, Hesam Fathi Moghadam, Ali Moharrer, Jingxiao Cai, Nikan Chavoshi, Venkatanathan Varadarajan, Sandeep R Agrawal, Sam Idicula, Tomas Karnagel, Sanjay Jinturkar, et al. 2020. Oracle automl: a fast and predictive automl pipeline. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3166–3180.
[81] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. 335–340.
[82] Richard Zhang and Daniel Golovin. 2020. Random Hypervolume Scalarizations for Provable Multi-Objective Black Box Optimization. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Hal Daumé III and Aarti Singh (Eds.), Vol. 119. PMLR, 11096–11105. https://proceedings.mlr.press/v119/zhang20i.html

A ADDITIONAL DETAILS ON BAYESIAN OPTIMIZATION

Bayesian optimization (BO) [3, 30, 32] is a sample-efficient strategy for exploring complex, high-dimensional search spaces—such as those encountered in ML pipeline tuning. BO is particularly well-suited for optimizing black-box, expensive, and multi-extremal objective functions [19, 47], providing the necessary flexibility to integrate stability into pipeline tuning. Its effectiveness is further supported by recent surveys [60] and benchmarks [44], which highlight its sample efficiency for tabular datasets.

BO relies on building a probabilistic approximation of the objective function **f**, often called a *surrogate model*, based on previously evaluated pipeline configurations. BO proceeds iteratively, balancing exploration and exploitation by incorporating two key components: (1) a *surrogate model* that captures our current belief about the performance landscape, and (2) an *acquisition function* that suggests the next most promising configuration to evaluate.

The *surrogate model* learns a posterior distribution over **f** by updating a prior belief using observations from prior evaluations. This posterior captures both the expected performance and the uncertainty associated with different regions of the search space [40, 64]. The *acquisition function* then leverages this posterior, typically combining its mean and variance, to prioritize candidate configurations that are either likely to perform well or lie in high-uncertainty regions, thus enabling efficient and informed exploration during the tuning process.

BO extends naturally to the multi-objective setting (MOBO) [19], where the goal is to efficiently approximate the *Pareto front* of trade-offs among multiple, often conflicting, objectives. MOBO maintains a probabilistic *surrogate model* for each objective, assuming independence, and uses an *acquisition function* to balance exploration and exploitation in selecting new configurations to evaluate. Since objective values are typically expensive to query and only known at specific points, MOBO emphasizes sample efficiency. Common strategies include *scalarization* [57, 82], which reduces multiple objectives to a single weighted sum; *hypervolume-based* methods, such as Expected Hypervolume Improvement (EHVI) [25], which aim to maximize the dominated region under the Pareto front; and *information-theoretic* approaches [11, 12, 68], which reduce uncertainty about the front. These techniques guide the search toward diverse, high-performing solutions with minimal evaluations.

Recent multi-objective Bayesian optimization (MOBO) methods [19, 58] have demonstrated competitive performance in incorporating fairness into the optimization process compared to other multi-objective hyperparameter optimization (HPO) techniques. Another advantage of BO is the availability of robust and efficient software frameworks [1, 6, 47], which provide standardized APIs to interact with different optimizers.

Multi-objective optimization (MO), unlike single-objective optimization, does not yield a unique optimal solution. Instead, it produces the *Pareto front* of dominant solutions, each representing a different trade-off among the objectives. `VirnyFlow` incrementally approximates the Pareto front during pipeline optimization.

Algorithm 2: NextLogicalPlan (NLP)

Input: Problem $\mathcal{P}$, dataset $\mathcal{D}$, exploration factor β, risk factor θ.
Output: Next *logical pipeline*.

```
if rand() < β then
    // Selection (Exploitation)
    Compute μ_k, δ_k and c_k for each logical pipeline k using the history
    LogicalPlan ← select a logical pipeline k with a probability proportional to μ_k + (θ/c_k) · δ_k
else
    // Random (Exploration)
    LogicalPlan ← random unseen logical pipeline
return LogicalPlan
```

B ADDITIONAL EXPERIMENTAL DETAILS

B.1 Datasets and Tasks

Tables 4-8 report the demographic composition of all datasets, specifically, the proportions and base rates of each protected group. The datasets are described in the following paragraphs.

`diabetes`[4] [74] was collected in India using a questionnaire comprising 18 questions covering aspects of health, lifestyle, and family background. It includes responses from 952 individuals, each described by 17 attributes — 13 categorical and 4 numerical — as well as a binary target variable indicating diabetes status. In this dataset, the sensitive attribute is *sex*, with "female" identified as the disadvantaged group.

Folktables [23] is another popular fairness dataset derived from US Census data from all 50 states between 2014-2018. The dataset has several associated tasks, of which we selected two: (i) ACSPublicCoverage (`folk-pubcov`) is a binary classification task to predict whether a low-income individual, not eligible for Medicare, has coverage from public health insurance. The dataset contains 19 features (17 categorical, 2 numerical) including disability, employment status, total income, and nativity. We use data from New York from 2018, subsampled to 50k rows. (ii) ACSEmployment (`folk-emp` and `folk-emp-big`) is a binary classification task to predict whether an individual is employed, from 16 features (15 categorical, 1 numerical) including educational attainment, employment status of parent, military status, and nativity. We use data from California from 2018, subsampled to 15k and 200k for `folk-emp` and `folk-emp-big`, respectively. In both tasks, *sex* and *race* are the sensitive attributes, with "female" and "non-White" as the disadvantaged groups.

`heart`[5] contains medical measurements related to cardiovascular conditions, covering 70,000 individuals. Each record includes 11 attributes — 6 categorical and 5 numerical — such as age, height, weight, and blood pressure, along with a binary target variable indicating the presence of heart disease. In this dataset, the sensitive attribute is *sex*, with "female" considered the disadvantaged group.

[4] https://www.kaggle.com/datasets/tigganeha4/diabetes-dataset-2019
[5] https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset

Table 4: Proportions and Base Rates for `diabetes` (≈1K).

	overall	gender_priv	gender_dis
Proportions	1.0	0.621	0.379
Base Rates	0.291	0.272	0.321

Table 5: Proportions and Base Rates for `folk-emp` (15K).

	overall	sex_priv	sex_dis	race_priv	race_dis	sex&race_priv	sex&race_dis
Proportions	1.0	0.487	0.513	0.627	0.373	0.804	0.196
Base Rates	0.571	0.621	0.524	0.563	0.586	0.578	0.544

Table 6: Proportions and Base Rates for `folk-pubcov` (50K).

	overall	sex_priv	sex_dis	race_priv	race_dis	sex&race_priv	sex&race_dis
Proportions	1.0	0.436	0.564	0.625	0.375	0.794	0.206
Base Rates	0.399	0.414	0.388	0.35	0.482	0.376	0.49

Table 7: Proportions and Base Rates for `heart` (70K).

	overall	gender_priv	gender_dis
Proportions	1.0	0.35	0.65
Base Rates	0.5	0.505	0.497

Table 8: Proportions and Base Rates for `folk-emp-big` (200K).

	overall	sex_priv	sex_dis	race_priv	race_dis	sex&race_priv	sex&race_dis
Proportions	1.0	0.49	0.51	0.625	0.375	0.806	0.194
Base Rates	0.569	0.616	0.524	0.562	0.58	0.576	0.54

B.2 Fairness Metrics for Comparison

To assess model fairness, we report error disparity metrics based on group-specific error rates, namely *True Positive Rate Difference (TPRD)*, *True Negative Rate Difference (TNRD)*, *False Negative Rate Difference (FNRD)*, and *Selection Rate Difference (SRD)*:

$$TPRD = \frac{TP_{dis}}{TP_{dis} + FN_{dis}} - \frac{TP_{priv}}{TP_{priv} + FN_{priv}}$$

$$TNRD = \frac{TN_{dis}}{TN_{dis} + FP_{dis}} - \frac{TN_{priv}}{TN_{priv} + FP_{priv}}$$

$$FNRD = \frac{FN_{dis}}{TP_{dis} + FN_{dis}} - \frac{FN_{priv}}{TP_{priv} + FN_{priv}}$$

$$SRD = \frac{TP_{dis} + FP_{dis}}{N_{dis}} - \frac{TP_{priv} + FP_{priv}}{N_{priv}}$$

B.3 Model Types

We evaluate predictive performance of 6 ML models in our experiments: (i) decision tree (`dt_clf`) with a tuned maximum tree depth, minimum samples at a leaf node, number of features used to decide the best split, and criteria to measure the quality of a split; (ii) logistic regression (`lr_clf`) with tuned regularization penalty, regularization strength, and optimization algorithm; (iii) light gradient boosted machine (`lgbm_clf`) with tuned number of boosted trees, maximum tree depth, maximum tree leaves, and minimum number of samples in a leaf; (iv) random forest (`rf_clf`) with a tuned number of trees, maximum tree depth, minimum samples required to split a node, and minimum samples at a leaf node; (v) extreme gradient boosting trees (`xgb_clf`) with a tuned tree depth, learning rate, number of boosting rounds, subsample ratio of the training instances, and minimum sum of instance weight needed in a child node; (vi) a deep table-learning method called GANDALF [41] (`gandalf_clf`) with a tuned learning rate, number of layers in the feature abstraction layer, dropout rate for the feature abstraction layer, and initial percentage of features to be selected in each Gated Feature Learning Unit (GFLU) stage. Search grids of hyperparameters for all models are defined in our codebase.

B.4 Baselines

We compare our system with two state-of-the-art, broad-spectrum AutoML baselines: `Alpine Meadow` [65] and `auto-sklearn` [29], both highlighted in recent benchmarks [33, 52] and surveys [7, 8]. `Alpine Meadow` (version 1.0.1) is an interactive AutoML tool that won the DARPA D3M competition in April 2019. It applies concepts from query optimization and introduces novel pipeline selection and pruning strategies using cost-based multi-armed bandits and Bayesian optimization (BO). We adopt some of these techniques to support experiment management in `VirnyFlow`, as discussed in Section 2.1. `auto-sklearn` (version 0.15.0) is a leading open-source AutoML system that tackles the CASH problem using Scikit-learn algorithms [16], incorporating BO, successive halving, ensembling, and meta-learning. It won the second ChaLearn AutoML challenge [35].

B.5 Computing Infrastructure

All experiments were conducted using a suitable experimental environment comprising a high-performance computing (HPC) cluster for execution and an Atlas M10 MongoDB replica set (3 data bearing servers with 2 vCPUs, 2 GB RAM, 1000 IOPS, up to 5 Gigabit network performance) for experiment management. We used the SLURM job scheduler to flexibly assign CPUs, RAM, and nodes for each job on the cluster. Unless stated otherwise, experiments with `VirnyFlow` using 32 workers were run on a single node with fixed resource allocations per dataset:

- `diabetes` (≈1K): 32 CPUs, 32 GB RAM
- `folk-emp` (15K): 32 CPUs, 64 GB RAM
- `folk-pubcov` (50K): 32 CPUs, 96 GB RAM
- `heart` (70K): 32 CPUs, 120 GB RAM
- `folk-emp-big` (200K): 32 CPUs, 150 GB RAM

The same resource configurations were used for `auto-sklearn`. `Alpine Meadow` required an additional 20 GB RAM per dataset compared to `VirnyFlow`. All computations were handled by a `Intel Xeon Platinum 8268 24C 205W 2.9GHz` processor and a `DDR4 2933MHz` RAM card.

All systems were implemented in isolated virtual environments using Python 3.9 and their respective libraries (listed in our repository), based on their original source code and run on Ubuntu 22.04. The code for `Alpine Meadow` was kindly provided by its authors, to whom we express our sincere gratitude. To launch Apache Kafka,

Table 9: Sensitivity of `VirnyFlow` to the training set fractions for pruning. Gray shadowing highlights the most optimal setting in terms of performance and efficiency.

Pruning	**Heart (70K)**		**Folk Pub. Cov. (50K)**	
	Score	Runtime	Score	Runtime
{1.0}	86.35 ±0.17	653 ±46	82.77 ±0.29	782 ±492
{0.25, 1.0}	86.33 ±0.20	939 ±632	82.64 ±0.31	765 ±146
{0.5, 1.0}	86.41 ±0.14	651 ±42	82.67 ±0.27	682 ±70
{0.75, 1.0}	86.35 ±0.20	795 ±166	82.78 ±0.44	789 ±120
{0.25, 0.5, 1.0}	86.18 ±0.60	815 ±142	82.91 ±0.43	970 ±205
{0.5, 0.75, 1.0}	86.36 ±0.08	766 ±111	82.74 ±0.33	871 ±205
{0.1, 0.25, 0.5, 1.0}	86.28 ±0.30	850 ±75	82.84 ±0.44	1104 ±166
{0.1, 0.5, 0.75, 1.0}	86.45 ±0.35	1403 ±1023	82.49 ±0.43	1007 ±81

we use Singularity containers[6], deployed on the same node before running an experiment. Specifically, we use the following Docker images: zookeeper — `docker://bitnami/zookeeper:3.9.3`, kafka brokers — `docker://bitnami/kafka:4.0.0`. Each experiment is repeated ten times to reduce the effect of randomness. All dependencies, hyperparameters, and system configurations are specified in our repository, along with installation and execution instructions in the README.

C ADDITIONAL EXPERIMENTAL RESULTS

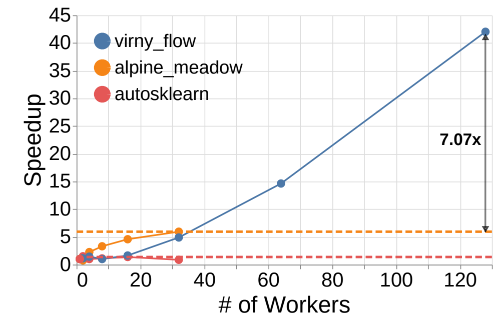
C.1 Scalability

Figures 6 and 5 present supplementary results on model accuracy and fairness for the scalability experiments using the `folk-emp-big` and `heart` datasets, as discussed in Section 3.4. These results further support the claim made in the main text that `VirnyFlow` consistently achieves higher *average scores* aggregated across multiple optimization objectives and equal objective weights. This trend holds not only for the smaller datasets examined in Section 3.3 but also for the larger datasets used in the scalability study.

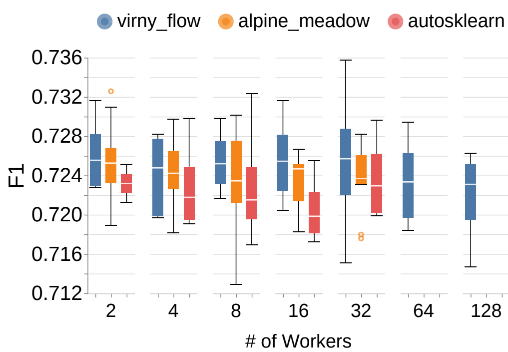
D VISUALIZATION INTERFACES

To enhance user interactivity, `VirnyFlow` integrates built-in visualization tools from `Virny` [37] and `OpenBox` [47]. `Virny` offers visualizations that help users explore trade-offs between different model performance metrics, both overall and across demographic groups. A demonstration of the `Virny` interface is available in the original paper [37] and through a Hugging Face web app[7]. `OpenBox` provides visualizations to monitor optimization progress and analyze pipeline tuning in detail. Figures 7, 8, and 9 show example plots from this visualization interface.

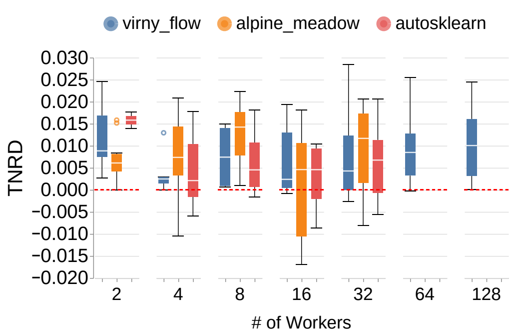
[6]https://docs.sylabs.io/guides/3.5/user-guide/introduction.html
[7]https://huggingface.co/spaces/denys-herasymuk/virny-demo



(a) Speedup

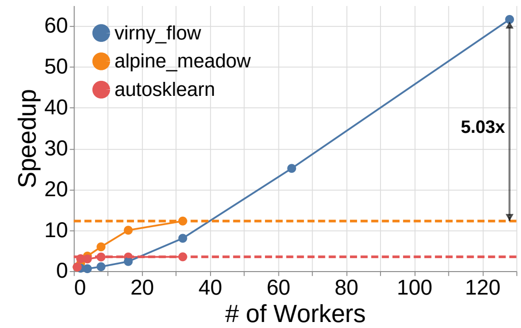


(b) F1

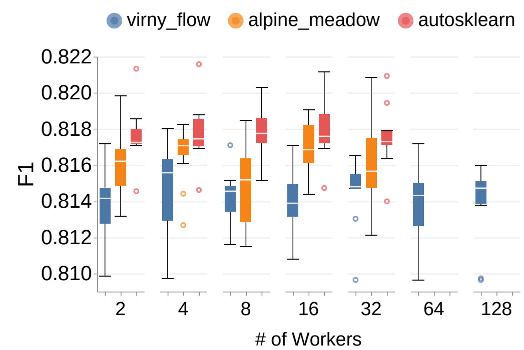


(c) TNRD

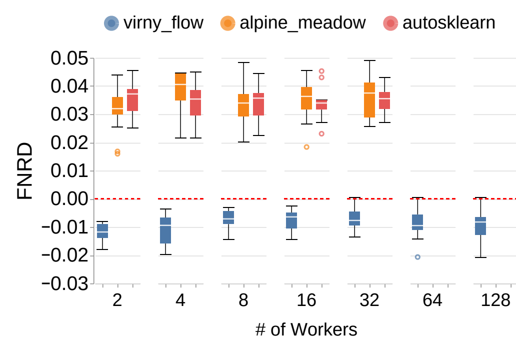
Figure 5: Scalability study on `heart` (70K). A dashed line highlights the best value for TNRD.



(a) Speedup



(b) F1



(c) FNRD

Figure 6: Scalability study on `folk-emp-big` (200K). A dashed line highlights the best value for FNRD.

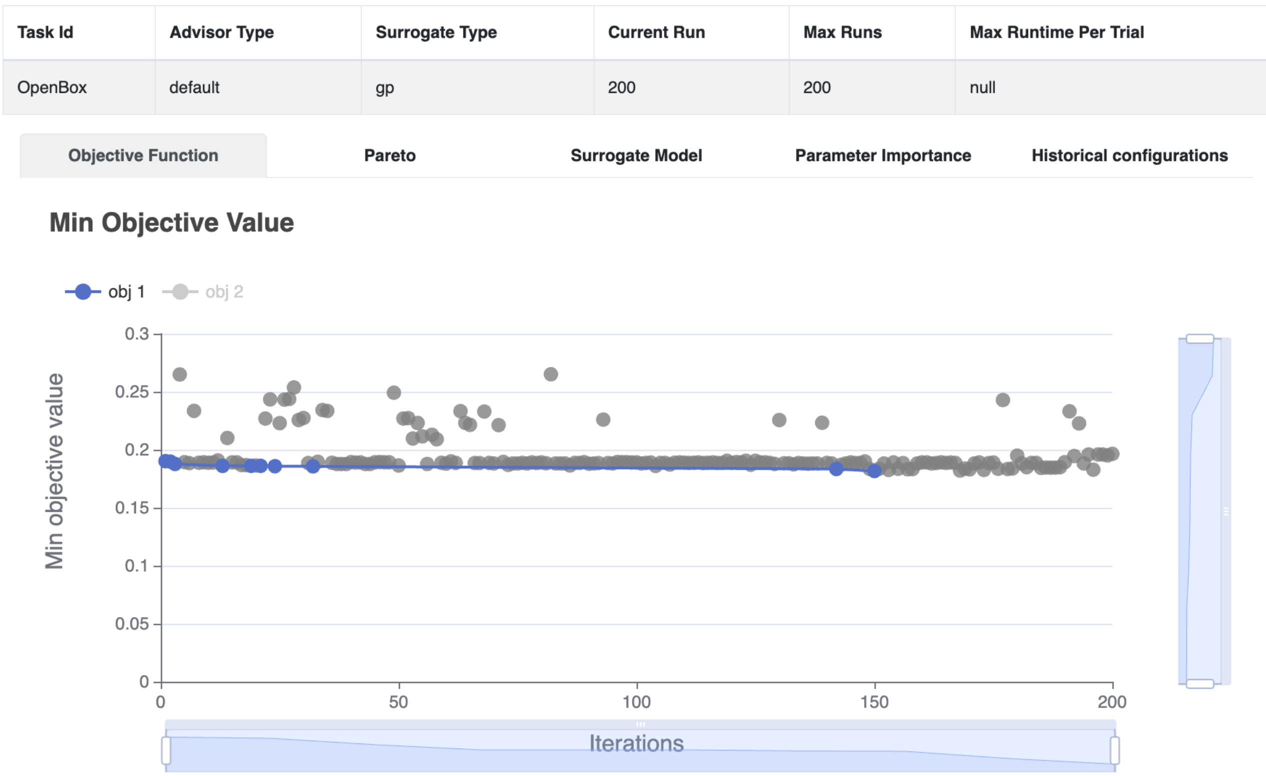


Figure 7: Optimization progress for Objective 1 (F1) for `folk-emp`. The x-axis shows the number of iterations. Since the MOBO optimizer in `VirnyFlow` minimizes the objective, all metrics are transformed accordingly. Lower values indicate better performance. A table at the top shows the current execution progress.

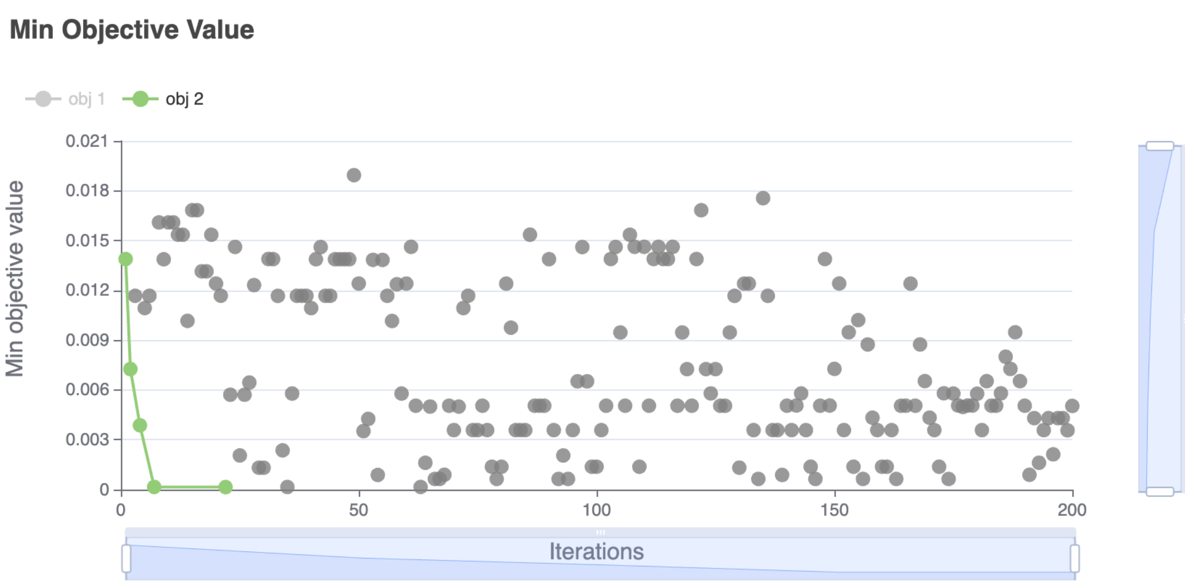


Figure 8: Optimization progress for Objective 2 (FNRD) for `folk-emp`. The x-axis shows the number of iterations. Since the MOBO optimizer in `VirnyFlow` minimizes the objective, all metrics are transformed accordingly. Lower values indicate better performance.

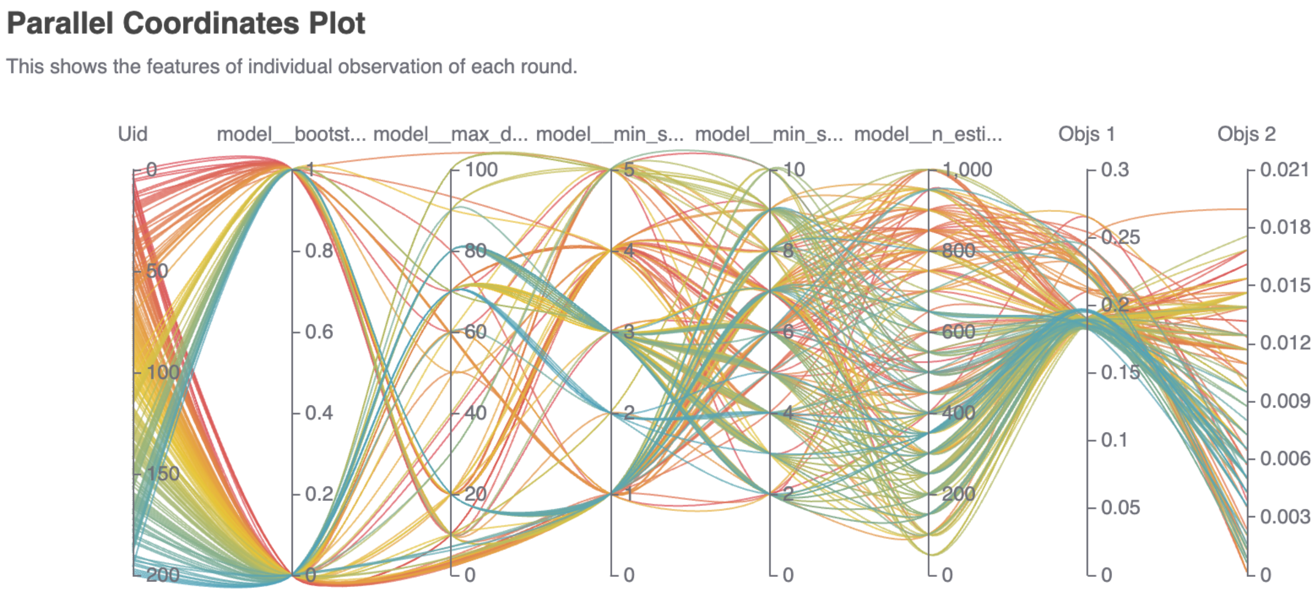


Figure 9: Parallel coordinates plot for `folk-emp`. A visualization shows used hyperparameters for the *logical pipeline* in each iteration and the respective outcome values of Objectives 1 and 2.