# Generating Skyline Datasets for Data Science Models*

Mengying Wang
Case Western Reserve University
Cleveland, Ohio, USA
mxw767@case.edu

Hanchao Ma
Case Western Reserve University
Cleveland, Ohio, USA
hxm382@case.edu

Yiyang Bian
Case Western Reserve University
Cleveland, Ohio, USA
yxb227@case.edu

Yangxin Fan
Case Western Reserve University
Cleveland, Ohio, USA
yxf451@case.edu

Yinghui Wu
Case Western Reserve University
Cleveland, Ohio, USA
yxw1650@case.edu

## ABSTRACT

Preparing high-quality datasets required by various data-driven AI and machine learning models has become a cornerstone task in data-driven analysis. Conventional data discovery methods typically integrate datasets towards a single pre-defined quality measure that may lead to bias for downstream tasks. This paper introduces **MODis**, a framework that discovers datasets by optimizing *multiple* user-defined, model-performance measures. Given a set of data sources and a model, **MODis** selects and integrates data sources into a skyline dataset, over which the model is expected to have the desired performance in all the performance measures. We formulate **MODis** as a multi-goal finite state transducer, and derive three feasible algorithms to generate skyline datasets. Our first algorithm adopts a "reduce-from-universal" strategy, that starts with a universal schema and iteratively prunes unpromising data. Our second algorithm further reduces the cost with a bi-directional strategy that interleaves data augmentation and reduction. We also introduce a diversification algorithm to mitigate the bias in skyline datasets. We experimentally verify the efficiency and effectiveness of our skyline data discovery algorithms, and showcase their applications in optimizing data science pipelines.

## 1 INTRODUCTION

High-quality machine learning (ML) models have become critical assets for various domain sciences research. A routine task in data-driven domain sciences is to prepare datasets that can be used to improve such data science models. Data augmentation [35] and feature selection [26] have been studied to suggest data for ML models [6]. Nevertheless, they typically generate data by favoring a pre-defined, single performance goal, such as data completeness or feature importance. Such data may be biased and not very useful to actually improve the model performance, and moreover, fall short at addressing multiple user-defined ML performance measures (*e.g.,* expected accuracy, training cost). Such need is evident in multi-variable experiment optimization [22, 29, 32], feature selection [26], and AI benchmarking [8], among others.

Discovering datasets that can improve a model over *multiple* user-defined performance measures remains to be desirable yet less studied issue. Consider the following real-world example.

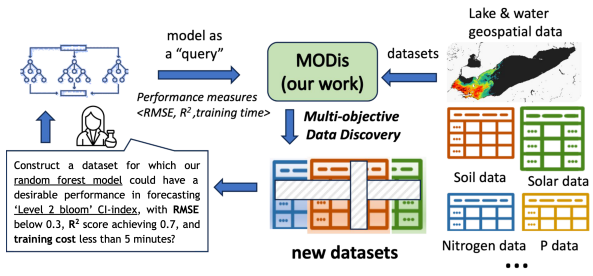



**Figure 1: Data generation for CI index prediction addressing multiple user-defined ML performance criteria, in order to improve an input ML model.**

**Example 1:** To assess the impact and causes of harmful algal blooms (HABs) in a lake, a research team aims to forecast the chlorophyll-a index (CI-index), a key measure of algal blooms. The team has gathered over 50 factors (*e.g.,* fertilizer, water quality, weather) of upstream rivers and watershed systems, and trained a random forest (RF) with a small, regional dataset. The team wishes to find new data with important spatiotemporal and chemical attributes, to generalize the RF model. In particular, the model is expected to perform well over such dataset in terms of three performance measures: root mean square error (*RMSE*), $R^2$ test, for "Level 2 bloom" CI-index, and training time cost. Desirably, the data generation process can inform *what* are crucial features to inspect, track *where* the feature values are from, and *how* they are integrated from the data sources.

The research team may issue a *skyline query* [3] that requests:

> "*Generate a dataset for which our random forest model for predicting 'Level 2 bloom' CI-index is expected to have a RMSE below* 0.3, $R^2$ *score at least* 0.7, *and incur a training cost in 5 minutes?*"

Here, the thresholds "0.3", "0.7", and "5 minutes" are set based on historical performance of the RF model over a data sample.

One may apply data integration, or perform feature engineering to refine existing datasets with important features. Nevertheless, these methods often fall short at consistently generate data towards optimizing user-defined ML performance, leaving alone the needs for addressing multiple measures *e.g.,* accuracy and training cost.

Another approach is to introduce a utility function as a linear weighted sum of multiple measures. This turns the need into a single objective. However, achieving both high accuracy and low training cost can be "conflicting"; moreover, a best dataset that optimizes such utility function may not necessarily satisfy the expected bounds as posed for each measure in the query.

Ideally, a data generation process should provide a dataset that ensures the model achieves best expected performance on

*The full version is available at http://arxiv.org/abs/2502.11262

at least one measure, with compromisingly good performance on the rest, and all satisfying the user-defined bounds if any. □

The above example calls for data generation approaches that can respond to the question by providing "skyline" datasets that can address multiple ML performance measures. More formally, given a query that specifies an input data science model $M$, a set of source tables $\mathcal{D} = \{D_1, \ldots D_n\}$, and a set of user-defined performance measures $\mathcal{P}$ (*e.g.,* accuracy, training time), our task is to generate a new table from $\mathcal{D}$, over which the expected performances of $M$ simultaneously reaches desirable goals for all measures in $\mathcal{P}$. As remarked earlier, traditional data integration and feature engineering with a pre-defined, single optimization objective falls short of generating data for such needs.

Moreover, a desirable data generation process should (1) declaratively produce such a data by simple, primitive operators that are well supported by established query engines and data systems, (2) perform data discovery without expensive model inference and validation; and (3) ensure quality guarantees on the resulting skyline dataset, for multiple performance measures. In addition, the generation should be efficient. This remains a challenging issue, considering large-scale data sources and the space of new datasets that can be generated from them.

**Contribution**. We introduce MODis, a multi-objective data discovery framework. MODis interacts data integration and ML model performance estimation to pursue a *multi-objective* data discovery paradigm. We summarize our contributions as follows.

(1) We provide a formal computation model for the skyline data generation process in terms of a finite state transducer (FST). An FST extends finite automata by associating an output artifact that undergoes modifications via sequences of state transitions. The formal model is equipped with (1) simple and primitive operators, and (2) a model performance oracle (Section 3). We use FST as an abstract tool to describe data generation algorithms and perform formal analysis to verify costs and provable quality guarantees.

(2) Based on the formal model, we introduce the skyline data generation problem, in terms of Pareto optimality (Section 4) . The goal is to generate a skyline set of datasets, ensuring each has at least one performance measure where the model's expected performance is no worse than any other dataset. While the problem is intractable, we present a fixed-parameter tractable result, for a polynomially bounded dataset exploration space from the running graph of an FST process, and a fixed measures set $\mathcal{P}$.

Based on the above formulation, we provide three feasible algorithms to generate skyline datasets.

(3) Our first algorithm provides an approximation on Pareto optimal datasets by exploring and verifying a bounded number of datasets that can be generated from data sources (Section 5.1). The algorithm adopts a "reduce-from-universal" strategy to dynamically drop values from a universal dataset towards a Pareto optimal set of tables. We show that this algorithm approximates Skyline set within a factor of $(1 + \epsilon)$ for all performance metric, and ensures exact dominance for at least one measure. In addition, we present a special case with a fully polynomial time approximation.

(4) Our second algorithm further reduces unnecessary computation. It follows a bi-directional scheme to prune unpromising data, and leverages a correlation analysis of the performance metrics to early terminate the search (Section 5.3).

(5) Moreover, we introduce a diversification algorithm to mitigate the impact of data bias (Section 5.4). We show that the algorithm achieves a $\frac{1}{4}$-approximation to an optimal diversified skyline dataset among all verified $(1 + \epsilon)$ counterparts.

Using real benchmark datasets and tasks, we experimentally verify the effectiveness of our data discovery scheme. We found that MODis is practical in use. For example, our algorithms take 30 seconds to generate new data, that can improve input models by 1.5-2 times in accuracy and simultaneously reduces their training cost by 1.7 times. It outperforms baseline approaches that separately performs data integration or feature selection; and remains feasible for larger datasets. Our case study also verified its practical application in domain science tasks.

**Related works**. We categorize related works as follows.

*Feature Selection.* Feature selection removes irrelevant and redundant attributes and identifies important ones for model training [26]. Filtering methods rank features in terms of correlation or mutual information [31, 34] and choose the top ones. They typically assume linear correlation among features, omitting collective effects from feature sets and hence are often limited to support directly optimizing model performance. Our method differs from feature selection in the following. (1) It generates skyline dataset with primitive data augmentation and reduction operators, beyond simply dropping the entire columns. (2) We generate data that improves the model over multiple ML performance measures, beyond retaining critical features; and (3) our method does not require internal knowledge of the models or incur learning overhead.

*Data Augmentation.* Data augmentation aims to create data from multiple data sources towards a unified view [6, 10, 35, 47]. It is often specified to improve data completeness and richness [35] and may be sensitive to the quality of schema. Our method aimes to generate data to improve the expected performance of data-driven models. This is different from the conventional data integration which mostly focuses on improving the data completeness. Generative data augmentation [5] synthesize new rows for multi-objective optimization with a predefined schema. In contrast, MODis generates data with both rows and columns manipulation. Also, HydraGAN requires a target column for each metric, while MODis supports user-defined metrics with configurable generation.

*Data Discovery.* Data discovery aims to prepare datasets for ML models [10, 13, 17, 24, 35]. For example, Kitana [17] computes data profiles (*e.g.,* MinHash) and factorized sketches for each dataset to build a join plan, and then evaluates the plan using a proxy model. METAM [13] involves the downstream task with a utility score for joinable tables. Comparing with prior work, we formalize data generation with cell-level operators, beyond joins. We target multi-objective datasets and provide formalization in terms of Pareto optimality. We also provide algorithms with quality guarantees and optimization techniques.

*Model Estimation.* Model estimation aims to provide accurate estimation of a model's performance without incurring expensive re-training and inference cost. For example, AutoML [19, 30, 44] train -surrogate models to estimate model performance [19, 30, 44], or predict the model performance by learning from past attempts [12] or Reinforcement Learning [9]. Model selection [41] leverages metadata and historical observations to build graph neural network-based estimator for estimating model performance. Our work leverage Multi-output Gradient Boosting as the surrogate model for fast and reliable estimation, and benefits

from established ML performance estimation approaches or other surrogate models.

## 2 MODELS AND PERFORMANCE EVALUATION

We start with several notations used in MODis framework.

**Datasets**. A dataset $D(A_1, \ldots A_m)$ is a structured table instance that conforms to a local schema $R_D(A_1, \ldots A_m)$. Each tuple $t \in D$ is a m-ary vector, where $t.A_i = a$ ($i \in [1, m]$) means the $i$th attribute $A_i$ of $t$ is assigned a value $a$. A dataset may have missing values at some attribute $A$ (*i.e.,* $t.A = \emptyset$).

Given a set of datasets $\mathcal{D} = \{D_1, \ldots D_n\}$, each dataset $D_i$ confirms to a local schema $R_i$. The *universal schema* $R_U$ is the union of the local schemas of datasets in $\mathcal{D}$, *i.e.,* a set of all the attributes involved in $\mathcal{D}$. The *active domain* of an attribute $A$ from $D_U$, denoted as $\mathsf{adom}(A)$, refers to the finite set of its distinct values occurring in $\mathcal{D}$. The size of $\mathsf{adom}(A)$, denoted as $|\mathsf{adom}(A)|$, is the number of distinct values of $A$ in $\mathcal{D}$.

**Models**. A data science model (or simply "model") is a function in the form of $M : D \rightarrow \mathbb{R}^d$, which takes as input a dataset $D$, and outputs a result embedding in $\mathbb{R}^d$ for some $d \in \mathbb{N}$. Here $\mathbb{R}$ and $\mathbb{N}$ are real and integer sets. In practice, $M$ can be a pre-trained machine learning model, a statistical model, or a simulator. The input $D$ may represent a feature matrix (a set of numerical feature vectors), or a tensor (from real-world physical systems), to be used for a data science model $M$ as training or testing data. The output embedding can be conveniently converted to task-dependent output (*e.g.,* labels for classification, discrete cluster numbers for clustering, or Boolean values for outlier detection) with post-processing.

*Fixed Deterministic models*. We say a model $M$ is *fixed*, if its computation process does not change for fixed input. For example, a regression model $M$ is fixed if any factors that determines its inference (*e.g.,* number of layers, learned model weights) remain fixed. The model $M$ is *deterministic* if it always outputs the same result for the same input. We consider fixed, deterministic models for the needs of consistent performance, which is a desired property in ML-driven data analytical tasks.

**Model Evaluation**. A *performance measure* $p$ (or simply "measure") is a performance indicator of a model $M$, such as accuracy *e.g.,* precision, recall, F1 score (for classification); or mean average error (for regression analysis). It may also be a cost measure such as training time, inference time, or memory consumption.

We use the following settings.

(1) We unify $\mathcal{P}$ as a set of normalized measures to *minimize*, with a range $(0, 1]$. Measures to be maximized (*e.g.,* accuracy) can be easily converted to an inversed counterpart (*e.g.,* relative error).

(2) Each measure $p \in \mathcal{P}$ has an optional range $[p_l, p_u] \in (0, 1]$. It specifies desired lower bound $p_l$ or an upper bound $p_u$ for model performance, such as maximum training or inference time, memory capacity, or error ranges.

**Remarks**. As we unify $\mathcal{P}$ as a set of measures to be minimized, it is intuitive that an upper bound $p_u$ specifies a "tollerence" for the estimated performance. We necessarily introduce a "lower bound" $p_l > 0$ for the convinience of (1) ensuring well-defined theoretical quality guarantees (as will be discussed in Section 5.1), and (2) leaving the option open to users for the configuration needs of downstream tasks such as testing, comparison or benchmarking.

| Symbol | Notation |
|---|---|
| $\mathcal{D}$, $D$, $D_U$ | a set of datasets, a single dataset, universal table |
| $R_D$, $R_U$ | local schema of $D$, and universal schema |
| $\mathcal{A}$, $A$, $\mathsf{adom}(A)$ | attribute set, attribute, and active domain |
| $M$ | a data science model $D \rightarrow \mathbb{R}^d$ |
| $\mathcal{P}$, $p$, $(p_l, p_u)$ | perform. measures, a measure, its range |
| $T$, $t = (M, D, \mathcal{P})$, $t.\mathcal{P}$ | test set; single test, its performance vector |
| $\mathcal{T} = (s_M, \mathcal{S}, \mathcal{O}, \mathcal{S}_F, \delta)$ | a data discovery system |
| $\mathcal{E}$ | a performance estimation model |
| $C = (s_M, \mathcal{O}, M, T, \mathcal{E})$ | a configuration of data discovery system |
| $G_{\mathcal{T}} = (\mathcal{V}, \delta)$ | running graph |
| $s \prec s'$, $D \prec D'$ | state dominance, dataset dominance |

**Table 1: Table of notations**

*Estimators*. A performance measure $p \in \mathcal{P}$ can often be efficiently estimated by an estimation model $\mathcal{E}$ (or simply "estimator"), in PTIME in terms of $|D|$ (the number of tuples in $D$). An estimator $\mathcal{E}$ makes use of a set of historically observed performance of $M$ (denoted as $T$) to infer its performance over a new dataset. It can be a regression model that learns from historical tuning records $T$ to predict the performance of $M$ given a new dataset $D$.

By default, we use a multi-output Gradient Boosting Model [33] that allows us to obtain the performance vector by a single call with high accuracy (see Section 6).

*Tests*. Given a model $M$ and a dataset $D$, a *test* $t$ is a triple $(M, D, \mathcal{P})$, which specifies a test dataset $D$, an input model $M$, and a set of user-defined measures $\mathcal{P} = \{p_1, \ldots p_l\}$. A test tuple $t = (M, D, \mathcal{P})$ is *valuated* by an estimator $\mathcal{E}$ if each of its measure $p \in \mathcal{P}$ is assigned a (estimated) value by $\mathcal{E}$.

**Example 2:** Consider Example 1. A pre-trained random forest (RF) model $M$ that predicts CI-index is evaluated by three measures $\mathcal{P} = \{\mathsf{RMSE}, \mathsf{R}^2, \mathsf{T}_{\mathsf{train}}\}$, which specifies the root mean square error, the $R^2$ score, and the training cost. A user specifies a desired normalized range of RMSE to be within $(0, 0.6]$, $\mathsf{R}^2$ in $[0, 0.35]$ *w.r.t.* a "inversed" lower bound $1 - 0.65$, and $\mathsf{T}_{\mathsf{train}}$ in $(0, 0.5]$ *w.r.t.* an upper bound of "3600 seconds" (*i.e.,* no more than 1800 seconds). □

We summarize the main notations in Table 1.

## 3 SKYLINE DATASET GENERATION: A FORMALIZATION

Given datasets $\mathcal{D}$, an input model $M$ and a set of measures $\mathcal{P}$, we formalize the generation process of a skyline dataset with a "multi-goals" *finite state transducer* (FST). An FST extends extends finite automata by associating outputs with transitions. We use FST to abstract and characterize the generation of Skyline datasets as a data transformation process. We introduce this formalization, with a counterpart for data integration [6, 25], to help us characterize the computation of skyline dataset generation.

**Data Generator**. A skyline dataset generator is a finite-state transducer, denoted as $\mathcal{T} = (s_M, \mathcal{S}, \mathcal{O}, \mathcal{S}_F, \delta)$, where (1) $\mathcal{S}$ is a set of states, (2) $s_M \in \mathcal{S}$ is a designated start state, (3) $\mathcal{O}$ is a set of operators of types $\{\oplus, \ominus\}$; (4) $\mathcal{S}_F$ is a set of output states; and (5) $\delta$ refers to a set of transitions. We next specify its components.

*States*. A *state* $s$ specifies a table $D_s$ that conforms to schema $R_s$ and active domains $\mathsf{adom}_s$. For each attribute $A \in R_s$, $\mathsf{adom}_s(A) \subseteq \mathsf{adom}(A)$ refers to a fraction of values $A$ can take at state $s$. $\mathsf{adom}_s(A)$ can be set as empty set $\emptyset$, which indicates that the attribute $A$ is not involved for training or testing $M$; or a wildcard '_' ('don't care'), which indicates that $A$ can take any value in $\mathsf{adom}(A)$.

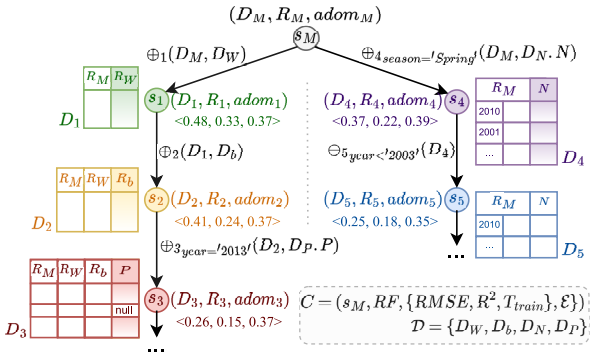



**Figure 2: A skyline data generation process, with a part of running graphs, and result datasets.**

*Operators.* A skyline data generator adopts two primitive polynomial-time computable operators, *Augment* and *Reduct*. These operators can be expressed by SPJ (select, project, join) queries, or implemented as user-defined functions (UDFs).

(1) *Augment* has a general form of $\oplus_c(D_M, D)$, which augments dataset $D_M$ with another $D \in \mathcal{D}$ subject to a literal $c$. Here $c$ is a literal in form of $A = a$ (an equality condition). An augmentation $\oplus_c(D_M, D)$ executes the following queries: (a) augment schema $R_M$ of $D_M$ with attribute $A$ from schema $R_D$ of $D$, *if* $A \notin R_M$; (b) augment $D_M$ with tuples from $D$ satisfying constraint $c$; and (c) fill the rest cells with "null" for unknown values.

(2) *Reduct* $\ominus_c(D_M)$: this operator(a) selects from $D_M$ the tuples that satisfy the selection condition posed by the literal $C$ posed on attribute $R_M.A$; and (b) removes all such tuples from $D_M$. Here $c$ is a single literal defined on $R_M.A$ as in (1).

*Transitions.* A *transition* $r = (s, op, s')$ is a triple that specifies a state $s = (D, R, \text{adom}_s)$, an operator op $\in \mathcal{O}$ over $s$, and a *result* state $s' = (D', R', \text{adom}'_s)$, where $D'$ and $R'$ are obtained by applying op over $D$ and $R$ conforming to domain constraint $\text{adom}'_s$, respectively.

In practice, the operators can be enriched by task-specific UDFs that perform additional data imputation, or pruning operations, to further improve the quality of datasets.

**Running**. A *configuration* of $\mathcal{T}$, denoted as $C = (s_M, \mathcal{O}, M, T, \mathcal{E})$, initializes a start state $s_M$ with a dataset $D_M$, a finite set of operators $\mathcal{O}$, a fixed deterministic model $M$, an estimator $\mathcal{E}$, and a test set $T$, where each test $t \in T$ has a valuated performance vector $\mathcal{F}(t)$. Both $D_M$ and $T$ can be empty set $\emptyset$. A *running* of $\mathcal{T}$ *w.r.t.* a configuration $C = (s_M, M, T, \mathcal{E})$ follows a general, deterministic process below.

(1) Starting from $s_M$, and at each state $s$, $\mathcal{T}$ iteratively applies operators from $\mathcal{O}$ to update a table with new attributes and tuples or mask its tuple values. This spawns a set of child states.

(2) For each transition $r = (s, \text{op}, s')$ spawned from $s$ with a result $s'$ by applying op, $\mathcal{T}$ (a) initializes a test tuple $t(M, D_{s'}, \mathcal{P})$ if $t \notin \mathcal{T}$ and invokes estimator $\mathcal{E}$ *at runtime* to valuate the performance vector of $t$; or (b) if $t$ is already in $T$, it directly loads $t.\mathcal{P}$.

Consistently, we say a state node $s \in \mathcal{V}$ is *valuated*, if a corresponding test $t(M, D_s, \mathcal{P})$ is valuated by $\mathcal{E}$. We denote its evaluated performance vector as $s.\mathcal{P}$.

The above process *terminates* at a set of output states $\mathcal{S}_F$, under an external termination condition, or no transition can be spawned (no new datasets can be generated with $\mathcal{O}$).

The *result* of a running $\mathcal{T}$ refers to the set of corresponding datasets $\mathcal{D}_F$ induced from the output states $\mathcal{S}_F$. As each output state $s \in \mathcal{S}$ uniquely determines a corresponding output dataset $D_s$, for simplicity, we shall use a single general term "output", denoted as $\mathcal{D}_F$, to refer to output states or datasets.

*Running graph.* A running of $\mathcal{T}$ can be naturally represented as the dynamic generation of a *running graph* $G_{\mathcal{T}} = (\mathcal{V}, \delta)$, which is a directed acyclic graph (DAG) with a set of state nodes $\mathcal{V}$, and a set of transition edges $r = (s, op, s')$. A *path* of length $k$ is a sequence of $k$ transitions $\rho = \{r_1, \ldots r_k\}$ such that for any $r_i = (s_i, op, s_{i+1})$, $r_{i+1} = (s_{i+1}, op, s_{i+2})$; *i.e.,* it depicts a sequence of transitions that converts an initial state $s_1$ with dataset $D_1$ to a result $s_k$ with $D_k$.

**Example 3:** Following Example 1, Fig 2 shows a fraction of a running graph with input set $\mathcal{D} = \{D_w, D_b, D_N, D_P\}$ (water, basin, nitrogen, and phosphorus tables, respectively). The augmentation $\oplus$ uses spatial joins [37], a common query that join tables with tuple-level spatial similarity. With a configuration $C = (s_M, \text{RF}, \{RMSE, R^2, T_{train}\}, \mathcal{E})$ (where $\mathcal{E}$ is an MO-GBM estimator), a running starts by joining $D_w$ and $D_b$ to get $D_2$. $D_2$ is then augmented with the attribute "`Phosphorus`" under a literal "`year` = *2013*", resulting in $D_3$ via a path $\{\oplus_1, \ldots, \oplus_3\}$. In each step, a test $t$ is initialized; and the estimator $\mathcal{E}$ is consulted to valuate the performance vector of $t$, and enrich $T$.

Consider another path from $D_M$ that results a dataset $D_5$ in Fig. 1. It first augmentes $D_M$ to $D_4$ with data in "Spring". A reduction with a condition "`year`<'2003" selects and removes all the tuples in $D_4$ with historical data before 2003, which leads to dataset $D_5$ that retains only the data since 2003. □

## 4 SKYLINE DATA GENERATION PROBLEM

Given $\mathcal{T}$ and a configuration $C$, MODis aims find a running of $\mathcal{T}$ that ideally leads to a "global" optimal dataset, where $M$ is expected to deliver the highest performance over all metrics. Nevertheless, a single optimal solution may not always exist. First, two measures in $\mathcal{P}$ may in nature conflict due to trade-offs (*e.g.,* training cost versus accuracy, precision versus recall). Moreover, the "no free lunch" theorem [38] indicates that there may not exist a single test that demonstrate best performance over all measures. We thus pursue *Pareto optimality* for $\mathcal{D}_F$. We start with a dominance relation below.

**Dominance**. Given a data discovery system $\mathcal{T}$ and performance measures $\mathcal{P}$, a state $s = (D_s, R_s, \text{adom}_s)$ is *dominated* by $s' = (D_{s'}, R_{s'}, \text{adom}_{s'})$, denoted as $s \prec s'$, if there are valuated tests $t = (M, D_s)$ and $t' = (M, D_{s'})$ in $T$, such that

- for each $p \in \mathcal{P}$, $t'.p \preceq t.p$; and
- there exists a measure $p^* \in \mathcal{P}$, such that $t'.p^* \prec t.p^*$.

A dataset $D_s$ is dominated by $D_{s'}$, denoted as $D_s \prec D_{s'}$, if $s \prec s'$.

*Skyline set.* Given $\mathcal{T}$ and a configuration $C$, let $\mathcal{D}_F$ be the set of all the possible output datasets from a running of $\mathcal{T}$, a set of datasets $\mathcal{D}_F^* \subseteq \mathcal{D}_F$ is a *skyline set w.r.t.* $\mathcal{T}$ and $C$, if

- for any dataset $D \in \mathcal{D}_F^*$, and any performance measure $p \in \mathcal{P}$, there exists a test $t \in T$, such that $t.p \in [p_l, p_u]$;
- there is no pair $\{D_1, D_2\} \subseteq \mathcal{D}_F^*$ such that $D_1 \prec D_2$ or $D_2 \prec D_1$; and
- for any other $D \in \mathcal{D}_F \setminus \mathcal{D}_F^*$, and any $D' \in \mathcal{D}_F^*$, $D \prec D'$.

We next formulate the skyline data generation problem.

**Skyline Data Generation**. Given a skyline data generator $\mathcal{T}$ and its configuration $C = (s_M, \mathcal{O}, M, T, \mathcal{E})$, the *skyline data generation* problem is to compute a skyline set $\mathcal{D}_F$ in terms of $\mathcal{T}$ and $C$.

**Example 4:** Revisiting prior example and consider the temporal results $\mathcal{D}_F$= $\{D_1, \ldots, D_5\}$ with the following performance vectors valuated by the estimator $\mathcal{E}$ so far:

| T: (D, M, $\mathcal{P}$, $\mathcal{E}$) | RMSE | $\hat{R^2}$ | $T_{train}$ |
|---|---|---|---|
| $t_1 : (D_1, \text{RF}, \mathcal{P}, \text{MO} - \text{GBM})$ | 0.48 | 0.33 | 0.37 |
| $t_2 : (D_2, \text{RF}, \mathcal{P}, \text{MO} - \text{GBM})$ | 0.41 | 0.24 | 0.37 |
| $t_3 : (D_3, \text{RF}, \mathcal{P}, \text{MO} - \text{GBM})$ | 0.26 | <u>0.15</u> | 0.37 |
| $t_4 : (D_4, \text{RF}, \mathcal{P}, \text{MO} - \text{GBM})$ | 0.37 | 0.22 | 0.39 |
| $t_5 : (D_5, \text{RF}, \mathcal{P}, \text{MO} - \text{GBM})$ | <u>0.25</u> | 0.18 | <u>0.35</u> |

Here $\hat{R^2}$ is inversed as 1-$R^2$: the smaller, the better. All the measures are normalized in $(0, 1]$ *w.r.t.* user-specified upper and lower bounds, and the optimal values are underlined. One can verify the following dominance relation among the datasets: (1) $D_1 \prec D_2 \prec D_3$, and $D_4 \prec D_5$; (2) $D_3 \not\prec D_5$ and vice versa. Hence a Skyline set $\mathcal{D}_F$ currently contains $\{D_3, D_5\}$. □

We present the following hardness result.

Theorem 1. *Skyline data generation is (1)* NP*-hard; and (2) fixed-parameter tractable, if (a) $\mathcal{P}$ is fixed, and (b) $|\mathcal{D}_F|$ is polynomially bounded by the input size $|\mathcal{D}|$.*

**Proof sketch:** The NP-hardness can be verified by a reduction from the Multiobjective Shortest Path problem (MOSP). Given an edge-weighted graph $G_w$, where each edge $e_w$ has a $d$-dimensional cost vector $e_w.c$, the cost of a path $\rho_w$ in $G_w$ is defined as $\rho_w.c = \sum_{e_w \in \rho_w} e_w.c$. The dominance relation between paths is determined by comparing their costs. MOSP is to compute a Skyline set of paths from start node $s$ to target node $t$.

Given an instance of MOSP, we construct an instance of our problem as follows. (1) We assign an arbitrally ordered index to the edges of $G_w$, say $e_1, \ldots e_n$. (2) We initialize a configuration $\mathcal{T}$ as follows. (a) $s_M$ has a single dataset $D_0$, where for each edge $e_i = (v, v')$, there is a distinct tuple $t_i \in D_0$. (b) $O$ contains a set of reduction operators, where each operator $o_i$ removes tuple $t_i$ from $D_0$, and incurs a pre-defined performance measure $e_i.c$. (c) $M$ maps each tuple $t_i$ in $D_0$ to a fixed embedding in $\mathbb{R}^d$. (d) The test set $T$ is $\emptyset$. We enforce the running graph of $\mathcal{T}$ to be the input $G_w$, by setting the initial state as $s$ with associated dataset $D_0$, a unique termination state as the node $t$, and the applicable transitions as the edges in $G_w$. One can verify that a solution of MOSP is a Pareto set of paths from $s$ to $t$, each results in a dataset by sequentially applying the reduction operators following the edges of the path. This yields a set of datasets that constitutes a corresponding skyline set $\mathcal{D}_F$ as a solution for our problem. As MOSP is shown to be NP-hard [15, 36], the hardness of skyline data generation follows.

To see the fixed-parameter tractability, we outline an exact algorithm. (1) The algorithm exhausts the runnings of a skyline generator $\mathcal{T}$, and invokes a PTIME inference process of the model $M$ and valuate at most $N \leq |\mathcal{D}_F|$ possible states (datasets). (2) It invokes a multi-objective optimizer such as Kung's algorithm [23]. This incurs $O(N \log N)^{|\mathcal{P}|-2}$ valuations when $|\mathcal{P}| \geq 4$, or $O(N(\log N))$ if $|\mathcal{P}| < 4$. As $N \leq |\mathcal{D}_F|$, and $|\mathcal{D}_F|$ is in $O(|D|)$, and $P$ is a fixed constant, the overall cost is in PTIME (see [42] for details). □

While the above exact algorithm is able to compute a skyline dataset, it remains infeasible even when enlisting $N$ states as a "once-for-all" cost is affordable. Moreover, a solution may contain an excessive number of datasets to be inspected. We next present three feasible algorithms, that generate datasets that approximate skyline sets with bounded size and quality guarantees.

# 5 COMPUTING SKYLINE SETS

## 5.1 Approximating Skyline Sets

We next present our first algorithm that generates a size-bounded set, which approximates a Skyline set in $\mathcal{D}_F$. To characterize the approximation quality, we introduce a notion of $\epsilon$-skyline set.

**$\epsilon$-Skyline set**. Given a data discovery system $\mathcal{T}$ with a configuration $C$, Let $\mathcal{D}_\mathcal{S}$ be a set of $N$ valuated datasets in the running of $\mathcal{T}$. Given a pair of datasets $(D, D')$ from $\mathcal{D}_\mathcal{S}$, and a constant $\epsilon > 0$, we say $D'$ *$\epsilon$-dominates* $D$, denoted as $D' \succeq_\epsilon D$, if for the corresponding tests $t = (M, D)$ and $t' = (M, D')$,

- $t'.p \leq (1 + \epsilon)t.p$ for each $p \in \mathcal{P}$, and
- there exists a measure $p^* \in \mathcal{P}$, such that $t'.p^* \leq t.p^*$.

In particular, we call $p^*$ a *decisive measure*. Note that $p^*$ can be any $p \in \mathcal{P}$ and may not be fixed.

A set of datasets $\mathcal{D}_\epsilon \subseteq \mathcal{D}_\mathcal{S}$ is an *$\epsilon$-Skyline set* of $\mathcal{D}_\mathcal{S}$, if

- for any dataset $D \in \mathcal{D}_\epsilon$, and any performance measure $p \in \mathcal{P}$, there exists a corresponding test $t \in T$, such that $t.p \in [p_l, p_u]$; and
- for every dataset $D' \in \mathcal{D}_\mathcal{S}$, there exists a dataset $D \in \mathcal{D}_\epsilon$ such that $D \succeq_\epsilon D'$.

**$(N, \epsilon)$-approximation**. We say an algorithm is an $(N, \epsilon)$-*approximation* for MODis, if it satisfies the following:

- it explores and valuates at most $N$ states;
- for any constant $\epsilon > 0$, the system correctly outputs an $\epsilon$-Skyline set, as an approximation of a Skyline set defined over $N$ *valuated* states; and
- the time cost is polynomial determined by $|\mathcal{D}|$, $N$, and $\frac{1}{\epsilon}$.

Below we present our main result.

Theorem 1. *Given datasets $\mathcal{D}$, configuration $C$, and a number $N$, there exists an $(N, \epsilon)$-approximation for* MODis *in $O\left(\min(N_u^{|R_u|}, N) \cdot \left(\left(\frac{\log(p_m)}{\epsilon}\right)^{|\mathcal{P}|-1} + I\right)\right)$ time, where $|R_u|$ is the size of the universal schema, $N_u = |R_u| + |\text{adom}_m|$ ($\text{adom}_m$ the largest active domain), $p_m = \max \frac{p_u}{p_l}$ as the measure $p$ ranges over $\mathcal{P}$; and $I$ is the unit valuation cost per test.*

**Remarks**. The above result captures a *relative* guarantee *w.r.t.* $\epsilon$ and $N$. When $N = \mathcal{D}_F$, an $(N, \epsilon)$-approximation ensures to output a $\epsilon$-Skyline set. The paradigm is feasible as one can explicitly trade the 'closeness' of the output to a Skyline set with affordable time cost, by explicitly tuning $\epsilon$ and $N$. Moreover, the worst-case factor $|\text{adom}_m|$ can also be "tightened" by a bound determined by the value constraints posed by the literals. For example, an attribute $A$ may contribute up to two necessary values in the search if the literals involving $A$ only enforce two equality conditions "A=2" and "A=5", regardless of how large $|\text{adom}(A)|$ is (see Sections 3 and 6).

## 5.2 Approximation Algorithm

As a constructive proof of Theorem 1, we next present an $(N, \epsilon)$-approximation algorithm, denoted as ApxMODis.

**"Reduce-from-Universal"**. Algorithm ApxMODis simulates the running of $\mathcal{T}$ from a start state $s_U$. The start state is initialized with a "universal" dataset $D_U$, which carries the universal schema $R_U$, and is populated by joining all the tables (with outer join to preserve all the values besides common attributes, by default). This is to enforce the search from a set of rows that preserve all the attribute values as much as possible to maximize the chance of

**Algorithm 1** :ApxMODis

**Input:** Configuration $C = (s_U, \mathcal{O}, M, T, \mathcal{E})$, a number $N$, a constant $\epsilon > 0$, a decisive measure $p_d$; user-specified upper bound $p_u$ for $p \in \mathcal{P}$;
**Output:** $\epsilon$-Skyline set $\mathcal{D}_F$.

**Queue** $Q := \emptyset$, integer $i := 0$, $Q$.enqueue($(s_U, 0)$);
**while** $Q \neq \emptyset$ **and** number of valuated states $< N$ **do**
  $(s, \mathrm{i}) := Q$.dequeue(); ${\mathcal{D}_F}^{i+1} := {\mathcal{D}_F}^{i}$;
  **for** each $(s', D_{s'}) \in$ OpGen $(s)$ **do**
    $Q$.enqueue($(s', \mathrm{i}+1)$);
    ${\mathcal{D}_F}^{i+1} :=$ UPareto $(s', {\mathcal{D}_F}^{i+1}, \epsilon)$;
**return** $\mathcal{D}_F$

**procedure** OpGen(s)
  **set** $Q' := \emptyset$;
  **for** each entry $l \in s.L$ **do**
    **if** $l = 1$ **then**
      $l := 0$;
      create a new state $s'$; $s'.L := s.L$;
      generate dataset $D_{s'}$ accordingly;
      $Q'$.append($(s', D_{s'})$);
  **return** $Q'$

**procedure** UPareto($s', {\mathcal{D}_F}^{i+1}, \epsilon$)
  update $s'.\mathcal{P}$ with estimator $\mathcal{E}$;
  **for** each $p \in \mathcal{P}$ **do**
    **if** $s'.\mathcal{P}(p) > p_u$ **then return** ${\mathcal{D}_F}^{i+1}$;
  update pos($s'$) with Equation (1);
  retrieve state $s''$ where pos($s''$) = pos($s'$);
  **if** no such $s''$ exists **then**
    ${\mathcal{D}_F}^{i+1} := {\mathcal{D}_F}^{i+1} \cup \{D'_s\}$;
  **else if** $s'.\mathcal{P}(p_d) < s''.\mathcal{P}(p_d)$ **then**
    ${\mathcal{D}_F}^{i+1} := {\mathcal{D}_F}^{i+1} \setminus \{D_{s''}\} \cup \{D_{s'}\}$;
  **return** ${\mathcal{D}_F}^{i+1}$

**Figure 3:** ApxMODis**: Approximating Skyline sets**

sequential applications of reduction only. It transforms state dominance into a "path dominance" counterpart. For a transition edge $(s, \ominus, s')$, a weight is assigned to quantify the gap between the estimated model performance over datasets $D_s$ and its "reduced" counterpart $D'_s$. The "length" of a path from $s_U$ to $s$ aggregates the edge weights towards the estimated performance of its result.

*Advantage.* We justify the "reduce-from-universal" strategy in the following context. (1) As the measures are to be minimized, we can extend "shortest" paths by prioritizing the valuation of datasets towards user-defined upper bounds with early pruning, to avoid unnecessary reduction. (2) Starting from a universal dataset allows early exploration of "dense" datasets, over which the model always tends to have higher accuracy in practice.

We next present the details of our algorithm.

**Auxiliary structure**. ApxMODis follows a dynamic levelwise state generation and valuation process, which yields a running graph $G_{\mathcal{T}}$ with up to $N$ nodes. It maintains the following.

(1) $Q$ is a queue that maintains the dynamically generated and valuated state nodes. Each entry of $Q$ is a pair $(s, i)$ that records a state and the level it resides.

(2) $\mathcal{D}_F$ is a list of datasets. $\mathcal{D}_F^i$ specifies the datasets processed at level $i$. Each state node $s$ is associate with a bitmap $L$ to encode if its schema $s.R_s$ contains an attribute $A$ in $D_U$, and if $D_s$ contains a value from its active domain adom$(A)$. The map is consulted to assert the applicability of reduct operators at runtime.

(3) Each state $s$ is associated with a position $pos(s)$ in a *discretized* $|\mathcal{P}-1|$-ary space, which is defined as

$$\mathrm{pos}(s) = \left[ \left\lfloor \log_{1+\epsilon} \frac{s.\mathcal{P}(p_1)}{p_{l_1}} \right\rfloor, \ldots, \left\lfloor \log_{1+\epsilon} \frac{s.\mathcal{P}(p_{|\mathcal{P}|-1})}{p_{l_{|\mathcal{P}|-1}}} \right\rfloor \right] \quad (1)$$

By default, we set the last measure in $\mathcal{P}$ as *a* decisive measure. We remark that one can choose any measure as decisive measure, and our results carry over.

**Algorithm**. The algorithm ApxMODis is illustrated in Fig. 3. It initializes a queue $Q$ with a start state $s_U$, and set a position to $s_U$ (line 4). In lines 5 to 9, it update the Skyline set $\mathcal{D}_F$ for each level iteratively. At level $d$, for each state $s \in Q$, procedure OpGen (line 12 to 19) explores all one-flip transitions in $s.L$ and generates a set with applicable reduct operators. ApxMODis enqueue new states and update the Skyline set ${\mathcal{D}_F}^{d+1}$ at next level accordingly by invoking Procedure UPareto. This process terminations until $N$ states are valuated, or no new state can be generated.

**Procedure** UPareto. Given a new state $s'$, procedure UPareto determines if $s'$ should be included in the current Skyline set. (1) It updates $s'.\mathcal{P}$ by consulting the estimator $\mathcal{E}$ (line 1), and decide an early skipping if its performance fails to satisfy the upperbound $p_u$ for some measure $p \in \mathcal{P}$. (2) Otherwise, UPareto updates the position of state $s'$, and decides if $s'$ "replaces" a valuated state $s''$ at the same position due to $(1+\epsilon)$-dominance (lines 6-9). The Skyline set at current level is updated accordingly with dataset $D_{s'}$.

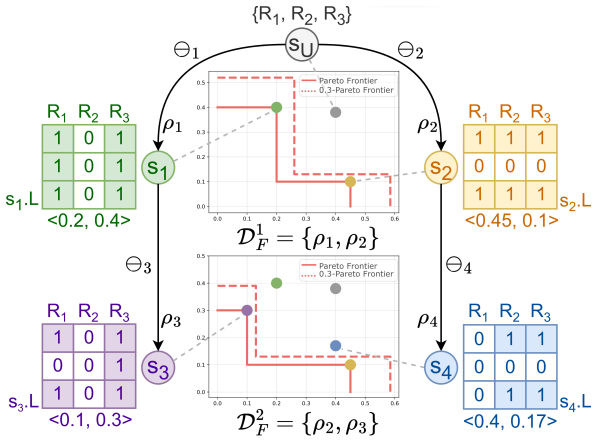



**Figure 4: "Reduct-from-Universal": an illustration of two-level computation. It performs multiple level-wise spawns and updates the $\epsilon$-Skyline set.**

**Example 5:** Fig. 4 illustrates data discovery of ApxMODis with $N$=5 and $\epsilon$=0.3, over a table set $\mathcal{D} = \{D_1, \ldots, D_3\}$ and measures $\mathcal{P} = < p_1, p_2 >$. The operator set $O$ contains four reduct operators $\{\ominus_1, \ldots \ominus_4\}$. (1) It first constructs a universal dataset $D_U$ with universal schema $R_U$. $D_U$ can be obtained by optimized multi-way join [45], augmentation [27], or UDFs [7]. The bitmap $D_U.L$ is initialized accordingly. Procedure OpGen then generates applicable reductions by "flipping" the entries in $D_U.L$. This spawns states $s_1$ and $s_2$ obtained by applying reduct $\ominus_1$ and $\ominus_2$, respectively It then consults the estimator $\mathcal{E}$ to valuate model performances, and identified that $D_1 \not\succeq_{0.3} D_2$ and vice versa. Thus ApxMODis sets the current 0.3-Skyline set as $\{D_1, D_2\}$.

ApxMODis next spawns states with applicable reductions $\ominus_3$ and $\ominus_4$, extending $\rho_1$ that leads to $s_1$, and $\rho_2$ that leads to $s_2$, This generates new extended paths $\rho_3$ and $\rho_4$ with results $D_3$ and $D_4$, respectively. It verfies that $\mathcal{D}_3 \succeq_{0.3} D_1$, but $D_2 \succeq_{0.3} D_4$; and $D_2 \not\succeq_{0.3} D_3$ and vice versa. This yields an updated 0.3-Skyline set $\{D_2, D_3\}$, after valuating 5 states. □

**Correctness & Approximability**. ApxMODis terminates as it spawns $N$ nodes with at most $|R_U||\text{adom}_m|$ distinct reduction, where $|\text{adom}_m|$ refers to the size of the largest active domain.

For approximability, we present the result below.

Lemma 2. *For any constant $\epsilon$,* ApxMODis *correctly computes an $\epsilon$-Skyline set $\mathcal{D}_F$ as an approximated Skyline set defined on the $N$ states it valuated.*

**Proof sketch:** We verify the $\epsilon$-approximability, with a reduction to the multi-objective shortest path problem (MOSP) [39]. Given an edge-weighted graph $G_w$, where each edge carries a $d$-dimensional attribute vector $e_w.c$, it computes a Skyline set of paths from a start node $u$. The cost of a path $\rho_w$ in $G_w$ is defined as $\rho_w.c = \sum_{e_w \in \rho_w} e_w.c$. The dominance relation between two paths is determined by the dominance relation of their cost. Our reduction (1) constructs $G_w$ as the running graph $G_{\mathcal{T}}$ with $N$ valuated state nodes and spawned transition edges; and (2) for each edge $(s, s')$, sets an edge weight as $e_w = s.\mathcal{P} - s'.\mathcal{P}$. Given a solution $\Pi_w$ of the above instance of MOSP, for each path $\rho_w \in \Pi$, we set a corresponding path $\rho$ in $G_{\mathcal{T}}$ with result dataset $D$, and adds it into $\mathcal{D}_F$. We can verify that $\Pi_w$ is an $\epsilon$-Skyline set of paths $\Pi_w$, if and only if $\mathcal{D}_F$ is an $\epsilon$-Skyline set of $\mathcal{D}_S$ that contains the datasets from the set of $N$ valuated states in $G_{\mathcal{T}}$. We then show that ApxMODis is an optimized process of the algorithm in [39], which correctly computes $\Pi_w$ for $G_w$. □

**Time cost**. Let $|R_u|$ be the total number of attributes in the universal schema $R_u$ of $D_u$, and $|\text{adom}_m|$ be the size of the largest active domain. ApxMODis performs $|R_u|$ levels of spawning, and at each node, it spawns at most $|R_u| + |\text{adom}_m|$ children, given that it "flips" one attribute for each reduction, and for each attribute, at most one domain value to mask. Let $N_u$ be $|R_u|+|\text{adom}_m|$. Thus, ApxMODis valuates at most $\min(N_u^{|R_u|}, N)$ nodes (datasets), taking $I \cdot \min(N_u^{|R_u|}, N)$ time, where $I$ refers to a polynomial time valuation cost of $\mathcal{E}$ per test. For each node, it then takes at most $\prod_{i=1}^{|\mathcal{P}|-1} \left(\left\lfloor \log_{1+\epsilon} \frac{p_{u_i}}{p_{l_i}} \right\rfloor + 1\right)$ time to update the $\epsilon$-Skyline set. Given $\epsilon$ is small, $\log(1+\epsilon) \approx \epsilon$, and the total cost is in $O\left(\min(N_u^{|R_u|}, N) \cdot \left(\left(\frac{\log(p_m)}{\epsilon}\right)^{|\mathcal{P}|-1} + I\right)\right)$ time. Given $|R_u|$ and $|\mathcal{P}|$ are small constants, the cost is polynomial in the input size $|D_u|$, $N$ and $\frac{1}{\epsilon}$. Theorem 1 thus follows.

**An FPTAS case**. We next present a case when ApxMODis ensures a stronger optimality guarantee. We say an $(N, \epsilon)$-approximation is a *fully polynomial time* approximation (FPTAS) for MODis, if (1) it computes an $\epsilon$-Skyline set for $\mathcal{D}_S$, where $\mathcal{D}_S$ refers to all possible datasets that can be generated from $D_U$, and (2) it runs in time polynomial in the size of $|D_U|$ and $\frac{1}{\epsilon}$.

Lemma 3. *Given a skyline dataset generator $\mathcal{T}$ with configuration $C$, if $|\mathcal{D}_S|$ has a size that is polynomially bounded in $O(f(|D_U|))$, then* ApxMODis *is an FPTAS for* MODis.

**Proof sketch:** We show this by a reduction from MODis to MOSP, similarly as in the approximability analysis. MOSP is known to have a fully polynomial time approximable (FPTAS) in terms of $\epsilon$-dominance. We set ApxMODis to run as a $(|\mathcal{D}_S|, \epsilon)$-approximation, which is a simplified implementation of an FPTAS in [39] with multiple rounds of "replacement" strategy following path dominance. As $|\mathcal{D}_S|$ is bounded by a polynomial of the input size $|D_U|$, it approximates a Skyline set for all in PTME. □

**Algorithm 2** :BiMODis

1: **Input:** Configuration $C = (s_U, \mathcal{O}, M, T, \mathcal{E})$, a constant $\epsilon > 0$;
2: **Output:** $\epsilon$-Skyline set $\mathcal{D}_F$.
3: **set** $s_b$ = BackSt($s_U$); **queue** $Q_f := \{(s_U, 0)\}$, $Q_b := \{(s_b, 0)\}$; integer $i := 0$;
4: **while** $Q_f \neq \emptyset$, $Q_b \neq \emptyset$ **and** $Q_f \cap Q_b = \emptyset$ **do**
5: $(s', i) = Q_f$.dequeue(); ▷ Forward Serach
6: $(s'', i) = Q_b$.dequeue(); ▷ Backward Serach
7: ${\mathcal{D}_F}^{i+1} = {\mathcal{D}_F}^{i}$;
8: **for all** $s^f \in$ OpGen $(s')$ **and** $s^b \in$ OpGen $(s'')$ **do**
9: ${\mathcal{D}_F}^{i+1} =$ UPareto $(s^f, {\mathcal{D}_F}^{i+1}, {\mathcal{D}_F}^{i}, \epsilon)$;
10: ${\mathcal{D}_F}^{i+1} =$ UPareto $(s^b, {\mathcal{D}_F}^{i+1}, {\mathcal{D}_F}^{i}, \epsilon)$;
11: **if** canPrune$(s^f, s^b)$ **then**
12: prune $(C, s^f, s^b)$;
13: $Q_f$.enqueue$((s^f, i+1))$, $Q_b$.enqueue$((s^b, i+1))$;
14: **return** $\mathcal{D}_F$

**Figure 5:** BiMODis**: Bi-directional Search**

The size bound of $\mathcal{D}_S$ is pragmatic and practical due to that the attributes often bear active domains that are much smaller than dataset size. Indeed, data science applications typically consider data with values under task-specific constraints. These suggest practical application of ApxMODis with affordable setting of $N$ and $\epsilon$. We present the detailed analysis in [42].

## 5.3 Bi-Directional Skyline Set Generation

Given our cost analysis, for skyline data generation with larger (more "tolerate") ranges $(p_l, p_u)$ and larger $|\mathcal{D}|$, ApxMODis may still need to valuate a large number of datasets. To further reduce valuation cost, we introduce BiMODis, its bi-directional variant. Our idea is to interact both augment and reduct operators, with a "forward" search from universal dataset, and a "backward" counterpart from a single dataset in $\mathcal{D}$. We also introduce a pruning strategy based on an early detection of dominance relation.

*Algorithm.* Algorithm BiMODis, as shown in Fig. 5, has the following steps. (1) *Initialization (lines 3).* It first invokes a procedure BackSt to initialize a back-end start state node $s_b$. Two queues $Q_f$ and $Q_b$ are initialized, seeded with start state $s_U$ for forward search, and a back state $s_b$ for backward search, respectively. They serve as the forward and backward frontiers, respectively. (2) *Bi-directional Search (lines 4-13).* BiMODis conducts an exploration from both directions, controlled by $Q_f$ for forward search, and $Q_b$ for backward search. Similar to ApxMODis, a Skyline set $\mathcal{D}_F$ is maintained in a levelwise manner. The difference is that it invokes a revised procedure OpGen (with original counterpart in ApxMODis in Fig. 3), which generates reduct operators for the forward search, and augment operators for the backward search. The search process terminates when both $Q_f$ and $Q_b$ are empty, or when a path is formed, the result $\mathcal{D}_F$ is returned.

*Procedure* BackSt. This procedure initializes a backend dataset $D_b$ for augmentation. This procedure can be tailored to the specific task. For example, for a classifier $M$ with input features and a

target attribute $A$ to be classified, we sample a small (minimal) set of tuples in $D_U$ to $D_b$ that covers all values of the active domain adom of $A$, to ensure that no classes will be "missed" in dataset $D_b$. Other task-specific strategies can also be applied here.

To reduce the valuation cost, BiMODis leverages correlation analysis over historical performance $T$, to assert "non-$\epsilon$-dominance" early, without a full valuation of their measures $\mathcal{P}$.

**Correlation-Based Pruning**. At runtime, BiMODis dynamically maintains a correlation graph $G_C$, where each node represents a measure in $\mathcal{P}$, and there is an edge $(p_i, p_j)$ in $G_C$ if $p_i$ and $p_j$ are *strongly correlated*, with an associated weight $|\text{corr}(p_i, p_j)|$ [46]. Here we say two measures are strongly correlated, if their Spearman correlation coefficient $\text{corr}(p_i, p_j) \geq \theta$, given their value distribution in the current set of tests $T$, for a user-defined threshold $\theta$. $G_C$ is dynamically updated, as more valuated tests are added to $T$.

*Parameterized Dominance*. BiMODis also "parameterize" any unvaluated measures in the performance vector $s.\mathcal{P}$ of a state $s$ with a potential range $[\hat{p_l}, \hat{p_u}] \subseteq [p_l, p_u]$. This range is derived from the valuated measures that are most strongly correlated, by consulting $G_C$ and test sets $T$. The entire vector $s.\mathcal{P}$ is incrementally updated, for each $p \in \mathcal{P}$, by setting (1) $s.\mathcal{P}(p)$ as $t.p$ (valuated), if there is a corresponding test $t = (M, D_s) \in T$ with $t.p$ valuated; or (2) $s.\mathcal{P}(p)$ as a variable with an estimated range $[s.\hat{p_l}, s.\hat{p_u}]$, if no test over $p$ of $D_s$ is valuated.

A state $s$ is *parameterized $\epsilon$-dominated by* another state $s'$, denoted as $s' \gtrapprox_\epsilon s$, if for each $p \in \mathcal{P}$,

- $s'.\mathcal{P}(p) \leq (1+\epsilon)s.\mathcal{P}(p)$, if both are valuated;
- $s'.\hat{p_u} \leq (1+\epsilon)s.\hat{p_l}$, if neither is valuated; or
- $s'.\mathcal{P}(p) \leq (1+\epsilon)s.\hat{p_l}$ (resp. $s'.\hat{p_u} \leq (1+\epsilon)s.\mathcal{P}(p)$), if $s'.\mathcal{P}(p)$ (resp. $s.\mathcal{P}(p)$) is valuated but $s.\mathcal{P}(p)$ (resp. $s'.\mathcal{P}(p)$) is not.

Based on the above construction, BiMODis monitors a monotonicity condition as follows.

*Monotonicity Condition*. Given the current test set $T$, we say a state $s$ (resp. $s'$) with a performance measure $p$ at a path $\rho$ has a *monotonicity property*, if for any state $s''$ reachable from $s$ (resp. can reach $s'$) via $\rho$, $s.\hat{p_u} < \frac{s''.\hat{p_l}}{1+\epsilon}$ (resp. $s'.\hat{p_u} < \frac{s''.\hat{p_l}}{1+\epsilon}$).

Given two states $s$ and $s'$, where $s' \gtrapprox_\epsilon s$, a state $s''$ on a path $\rho$ from $s$ or to $s'$ *can be pruned* by Correlation-based Pruning, if for every $p \in \mathcal{P}$, $s''$ has $p$ at $\rho$ with a monotonicity property *w.r.t.* $s$ (resp. $s'$). We present the following pruning rule.

Lemma 4. *Let $s \in Q_f$ and $s' \in Q_b$. If $s' \gtrapprox_\epsilon s$, then for any state node $s''$ on a path from $s$ or to $s'$ that can be pruned by Correlation-Based Pruning, $D_{s''}$ is not in any $\epsilon$-Skyline set of the datasets that can be generated from valuated states.*

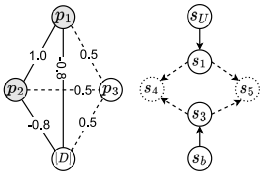
**Proof sketch:** We verify the pruning rule with a case study of $s$ and $s'$, subject to the monotonicity property. **Case 1: Both $s'.\mathcal{P}(p)$ and $s.\mathcal{P}(p)$ are valuated.** If $s' \gtrapprox_\epsilon s$, then by definition, $s'.\mathcal{P}(p) \leq (1+\epsilon)s.\mathcal{P}(p)$ for all $p \in \mathcal{P}$. This readily leads to $\epsilon$-dominance, *i.e.,* $s' \succeq_\epsilon s$. As $s''$ has every performance measures $p \in \mathcal{P}$ with a monotonicity property *w.r.t.* $s$, $s \succeq_\epsilon s''$. Hence $s''$ can be safely pruned without valuation. **Case 2: Neither $s'.\mathcal{P}(p)$ nor $s.\mathcal{P}(p)$ is valuated.** By definition, as $s' \gtrapprox_\epsilon s$, then for every $p \in \mathcal{P}$, $s'.\hat{p_u} \leq (1+\epsilon)s.\hat{p_l}$. Given that $s''$ has every performance measures $p \in \mathcal{P}$ with a monotonicity property *w.r.t.* $s$, then by definition, for each $p \in \mathcal{P}$, we have $s'.p \leq s'.\hat{p_u} \leq (1+\epsilon)s.\hat{p_l} \leq (1+\epsilon)s.\hat{p_u} < (1+\epsilon)\frac{s''.\hat{p_l}}{1+\epsilon} \leq s''.p$, for every $p \in \mathcal{P}$. By definition of state dominance, $s' \succ s''$, for unevaluated $s''$. Following a similar proof, one can infer that $s \succ s''$ for a state $s$ in the forward front of BiMODis. Hence $s''$ can be safely pruned. **Case 3: One of $s'.\mathcal{P}(p)$ or $s.\mathcal{P}(p)$ is valuated.** Given that $s' \gtrapprox_\epsilon s$, we have (a) $s'.\mathcal{P}(p) \leq (1+\epsilon)s.\hat{p_l}$, if only $s'.\mathcal{P}(p)$ is valuated; or (b) $s'.\hat{p_u} \leq (1+\epsilon)s.\mathcal{P}(p)$, if only $s.\mathcal{P}(p)$ is valuated. Consider case 3(a). As $s$ can reach $s''$ via a path $\rho$, and $s''$ satisfiies the pruning condition, we can infer that $s'.\mathcal{P}(p) \leq (1+\epsilon)s.\hat{p_l} \leq (1+\epsilon)s.\hat{p_u} < (1+\epsilon)\frac{s''.\hat{p_l}}{1+\epsilon} \leq s''.p$, hence $s' \succ s''$. Similarly for case 3(b), we can infer that $s'.\hat{p_u} \leq (1+\epsilon)s.p \leq (1+\epsilon)s.\hat{p_u} < (1+\epsilon)\frac{s''.\hat{p_l}}{1+\epsilon} \leq s''.p$. hence $s' \succ s''$. For both cases, $s''$ can be pruned without evaluation. Lemma 4 hence follows. □

Procedures canPrune and prune (lines 11-12; omitted) asserts the Correlation-Based Pruning condition, and perform the maintenance of $G_C$, $T$ and other auxiliary structures, respectively. Note that the above rule is checkable in PTIME *w.r.t.* input size $|\mathcal{D}_S|$. When $|\mathcal{D}_S|$ is large, one can generate a path with its states unevaluated, and check at runtime if the condition holds between evaluated states in the forward and backward frontier.

We present the detailed analysis in [42].

**Example 6:** We illustrate *Correlation-Based Pruning* in the figure below. From left to right, it depicts a set of test records $R$, the correlation graph $G_C$, and part of the running graph $G_\mathcal{T}$. $G_C$ is constructed from $T$ with measures as nodes and Spearman correlations as edge weights. For each $s_n \in G_\mathcal{T}$, the associated $p \in \mathcal{P}_{s_n}$ is obtained by test $t_{s_n} = (M, D_{s_n})$.

| | Label | $p_1$ | $p_2$ | $p_3$ |
|---|---|---|---|---|
| $s_U$ | (1, 1, 1, 1) | 0.42 | 0.18 | 0.9 |
| $s_1$ | (1, 1, 1, 0) | 0.4 | 0.17 | 0.1 |
| $s_2$ | (1, 0, 0, 1) | 0.5 | 0.22 | / |
| $s_3$ | (0, 1, 0, 0) | **0.45** | / | / |
| $s_b$ | (0, 0, 0, 0) | 0.6 | 0.4 | 0.3 |




At $\theta = 0.8$, $p_1$ and $p_2$ are positively correlated with each other and negatively correlated with $|D|$, so $\mathcal{P}^k = \{p_1, p_2\}$. From $s_U$ and $s_b$, the forward and backward frontiers derive states $s_1$ and $s_3$, respectively. To estimate $s_3.\mathcal{P}(p_2)$, note that $s_3.\mathcal{P}(p_1) = 0.45$, which lies between $s_U.\mathcal{P}(p_1) = 0.42$ and $s_2.\mathcal{P}(p_1) = 0.5$. Given the strong correlation between $p_1$ and $p_2$, we infer $s_3.\mathcal{P}(p_2)$ to be within the interval $[0.18, 0.22]$, with $\hat{p_{l_2}} = s_U.\mathcal{P}(p_2)$ and $\hat{p_{u_2}} = s_2.\mathcal{P}(p_2)$. With $\epsilon = 0.3$, we find $s_3 \gtrapprox_\epsilon s_1$ because $0.45 \leq (1+0.3) \cdot 0.4$ and $0.22 \leq (1+0.3) \cdot 0.17$. For intermediate states $s_4$ (bitmap entry $(1, 1, 0, 0)$) and $s_5$ (bitmap entry $(0, 1, 1, 0)$), which are not recorded in $T$ and have $|D_{s_4}| = |D_{s_5}| = 2$, a similar inference process shows they fall within the bounds set by $[s_1.\mathcal{P}, s_3.\mathcal{P}]$. As a result, $s_4$ and $s_5$ can be pruned. □

**Time Cost**. BiMODis takes the same time complexity as ApxMODis. The $(N, \epsilon)$-approximation holds for BiMODis given that it correctly updates the $\epsilon$-Skyline set by definition. Our experimental study verifies that it is much faster in practice and particularly suitable for larger $\epsilon$ or search spaces (represented by maximum path length). It also scales more efficiently for large datasets (see Exp-3 in Section 6).

## 5.4 Diversified Skyline Dataset Generation

A Skyline dataset may still contain data that largely overlap or are similar, hence leading to bias and reducing the generality of the model if adopted. This may occur due to skewed value distribution in the active domains, common attributes, over specific performance metrics in the skyline data generation process. It is

**Algorithm 3** :Diversification step at level $i$

**Input:** $\epsilon$-Skyline set ${\mathcal{D}_F}^i$ (from UPareto), integer $k$;
**Output:** a diversified $k$-subset of ${\mathcal{D}_F}^i$ (to be passed to level $i+1$).
**if** $|{\mathcal{D}_F}^i| \leq k$ **then return** ${\mathcal{D}_F}^i$
initialize $\mathcal{D}_F^P$ with $k$ random dataset in ${\mathcal{D}_F}^i$;
score := $div(\mathcal{D}_F^P)$;
**for all** $D \in \mathcal{D}_F^P$ **do**
  **for all** $D' \in {\mathcal{D}_F}^i$ **do**
    **if** $D' \in \mathcal{D}_F^P$ **then** Continue;
    $\mathcal{D}_F^{P'} := (\mathcal{D}_F^P \setminus \{D\}) \cup \{D'\}$;
    score' := $div(\mathcal{D}_F^{P'})$
    **if** score' > score **then**
      $\mathcal{D}_F^P := \mathcal{D}_F^{P'}$, score := score';
**return** $\mathcal{D}_F^P$

**Figure 6: Level-wise diversification of** DivMODis

often desirable to explore a diversified variant of skyline set generation to create varied datasets that mitigate such bias [22, 29].

Given $\mathcal{T}$ and a configuration $C$, a set of datasets $\mathcal{D}$, constants $N$, $\epsilon$ and $k$, the *diversified skyline data generation* is to compute a set $\mathcal{D}_F^*$ of at most $k$ tables, such that (1) $\mathcal{D}_F^*$ is an $\epsilon$-Skyline set of $N$ valuated states by an $(N, \epsilon)$-approximation of MODis, and (2) among all $\epsilon$-Skyline sets over $N$ states valuated in (1), it maximizes a diversification score defined as:

$$\mathsf{div}(\mathcal{D}_F) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \mathsf{dis}(D_i, D_j) \tag{2}$$

where a distance function dis quantifies the difference of datasets in terms of both value distributions and estimated performance, and is defined as:

$$\mathsf{dis}(D_i, D_j) = \alpha \frac{1 - \cos(s_i.L, s_j.L)}{2} + (1-\alpha)\frac{\mathsf{euc}(t_i.\mathcal{P}, t_j.\mathcal{P})}{\mathsf{euc_m}}$$

We adopt Cosine similarity cos and Euclid Distance (euc). The latter is normalized by the maximum Euclid Distance $euc_m$ among the historical performances in $T$.

We next outline an algorithm, denoted as DivMODis, that extends an $(N, \epsilon)$-approximation to computes an a diversified $\epsilon$-Skyline set $\mathcal{D}_F$ of at most $k$ datasets.

*Algorithm.* DivMODis revises MODis by incrementally diversify an input $\epsilon$-Skyline set ${\mathcal{D}_F}^i$ at level $i$ (partially shown in Fig.6). It derives $\mathcal{D}_F^P$ by a greedy selection and replace strategy as follows. (1) It initializes $\mathcal{D}_F^P$ as a random $k$-set from ${\mathcal{D}_F}^i$, and updates $\mathcal{D}_F^P$ by incrementally replacing tables with the highest marginal gain in diversification, hence an improved $div(\mathcal{D}_F^P)$. (2) $\mathcal{D}_F^P$ is passed to be processed at level $i+1$, upon the arrival of new states. DivMODis returns the diversified set $\mathcal{D}_F$, following the same termination condition as in ApxMODis.

We show that the diversified MODis can be approximated, for a submodular diversification function div. Our result holds for the specification of div in Equation 2.

Lemma 5. *Given $N$ and $\epsilon$,* DivMODis *achieves a $\frac{1}{4}$ approximation for diversified* MODis, i.e., *(1) it correctly computes a $\epsilon$-Skyline set $D_F^P$ over $N$ valuated datasets, and (2)* $\mathsf{div}(D_F^P) \geq \frac{1}{4}\mathsf{div}(\mathcal{D}_\mathsf{F}^*)$.

**Proof sketch:** We show an induction on the levels. (1) We verify the guarantee at a single level, by constructing an approximation preserving reduction to the stream submodular maximization problem [2]. Given a stream $E = \{e_0, \ldots e_m\}$, an integer $k$, and a submodular diversification function $f$, it computes a $k$-set of elements $S$ that can maximize $f(S)$. Our reduction constructs a stream of datasets $\mathcal{D}_S$ following the level-wise generation. We show that the function div is a submodular function. (2) By integrating a greedy selection and replacement policy, DivMODis keeps a $k$-set with the most diverse and representative datasets to mitigate the biases in the Skyline set. DivMODis achieves a $\frac{1}{4}$-approximation of an $\epsilon$-Skyline set with maximized diversity at each level $i$. Please see the detailed proof in [42]. □

| Dataset Sets | # tables | # Columns | # Rows |
|---|---|---|---|
| Kaggle | 1943 | 33573 | 7317K |
| OpenData | 2457 | 71416 | 33296K |
| HF | 255 | 1395 | 10207K |

**Table 2: Characteristics of Datasets**

**Analysis**. DivMODis incurs an overhead to update the diversified $k$-set. As MODis valuates up to $\min(N_u^{|R_u|}, N)$ nodes (datasets), the total additional overhead is in $O(\min(N_u^{|R_u|}, N) \cdot k \cdot T_\mathcal{S})$, where $T_\mathcal{S}$ refers to the unit valuation cost for a single table, which is in PTIME. As both $k$ and $T_\mathcal{S}$ are relatively small, the practical overhead for DivMODis remains small (see Sec. 6).

**Remarks**. Alternatives that solve multi-objective optimization may be applied, such as evolutionary algorithms such as NSGA-II [4], or reinforcement-learning based methods [28]. The former rely on costly stochastic processes (e.g., mutation and crossover) and may require extensive parameter tuning. The latter are effective for general state exploration but require high-quality training samples and may not converge over "conflicting" measures. In contrast, MODis is training and tuning free. Our experiments verified that ApxMODis provides early generation of high-quality datasets from a few large input datasets, due to "reduce-from-Universal" strategy, BiMODis enhances efficiency through bidirectional exploration and pruning, hence benefits for larger number of small-scale datasets, and DivMODis benefits most for datasets with skewed distribution.

## 6 EXPERIMENT STUDY

We next experimentally verify the efficiency and effectiveness of our algorithms. We aim to answer three questions: **RQ1**: How well can our algorithms improve the performance of models in multiple measures? **RQ2**: What is the impact of generation settings, such as data size? **RQ3**: How fast can they generate skyline sets, and how scalable are they? We also illustrate the applications of our approaches with case studies[1].

**Datasets.** We use three sets of tabular datasets: kaggle [20], OpenData [1], and HF [18] (summarized in Table 2).

**Tasks and Models.** A set of tasks are assigned for evaluation. We trained: (1) a Gradient Boosting model (GBmovie) to predict movie grosses using Kaggle for Task $T_1$; (2) a Random Forest model (RFhouse) to classify house prices using OpenData with the same settings in [13] for Task $T_2$; and (3) a Logistic Regression model (LRavocado) to predict Avocado prices using HF for Task $T_3$. (4) a LightGBM model (LGCmental) [21] to classify mental health status using Kaggle for Task $T_4$. We also introduced task $T5$, a link regression task for recommendation. This task takes as input a bipartite graph between users and products, and links indicate their interaction. A LightGCN [16] (LGRmodel), a variant

[1]Our codes and datasets are available at github.com/wang-mengying/modis

| Notation | Measures | Used In |
|---|---|---|
| $p_{Acc}$ | Model Accuracy | $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_4$ |
| $p_{Tr}$ | Training Time Cost | $\mathcal{P}_1$-$\mathcal{P}_4$ |
| $p_{F1}$ | $F_1$ score | $\mathcal{P}_2$, $\mathcal{P}_4$ |
| $p_{AUC}$ | Area under the curve | $\mathcal{P}_4$ |
| $p_{Nc(n)}$ | NDCG(@n) | $\mathcal{P}_5$ |
| $p_{MAE}$, $p_{MSE}$ | Mean Absolute / Squared Error | $\mathcal{P}_3$ |
| $p_{Pc(n)}$, $p_{Rc(n)}$ | Precision(@n), Recall(@n) | $\mathcal{P}_5$ |
| $p_{Fsc}$ | Fisher Score [26] | $\mathcal{P}_1$, $\mathcal{P}_2$ |
| $p_{MI}$ | Mutual Information [13, 26] | $\mathcal{P}_1$, $\mathcal{P}_2$ |

**Table 3: Performance Measures**

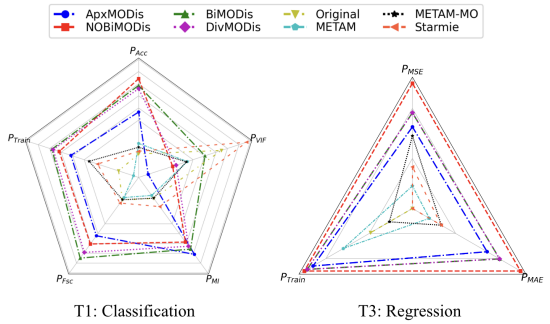



**Figure 7: Effectiveness: Multiple Measures**

of graph neural networks (GNN) optimized for fast graph learning, is trained to predict top-$k$ missing edges in an input bipartite graph to suggest products to users. A set of 1873 bipartite graphs is constructed from Kaggle for $T_5$. The "augment" (resp. "reduct") operators are defined as edge insertions (resp. edge deletions) to transform a bipartite graph to another.

We use the same training scripts for each task and all methods for a fair comparison. We assigned measures $\mathcal{P}_1$ through $\mathcal{P}_5$ for tasks $T_1$ to $T_5$, respectively , as summarized in Table 3. We also report the size of the data ($p_{DSize}$) in terms of (total # of rows total # of columns), excluding attributes with all cells masked.

*Estimator* $\mathcal{E}$. We adopt MO-GBM [33] as a desired model performance estimator. It outperforms other candidate models even with a simple training set For example, for $T_1$, MO-GBM performs inference for all objectives on one state in at most 0.2 seconds, with a small MSE of 0.0003 when predicting "Accuracy".

**Algorithms**. We implemented the following methods.

(1) **MODis**: Our multi-objective data discovery algorithms, including ApxMODis, BiMODis, and DivMODis. We also implemented NOBiMODis, a counterpart of BiMODis without correlation-based pruning. (2) METAM [13]: a goal-oriented data discovery algorithm that optimizes a single utility score with consecutive joins of tables. We also implemented an extension METAM-MO, by incorporating multiple measures into a single linear weighted utility function. (3) Starmie [11]: a data discovery method that focuses on table-union search and uses contrastive learning to identify joinable tables. For METAM and Starmie, we used the code from original papers. (4) SkSFM [33]: An automated feature selection method in scikit-learn's SelectFromModel, which recommends important features with a built-in estimator. (5) H2O [14]: an AutoML platform; we used its feature selection module, which fits features and predictors into a linear model.

**Construction of $D_U$ and Operators**. To prepare universal datasets $D_U$ for MODis, we preprocess Kaggle, OpenData and HF into joinable tables and construct $D_U$ with multi-way joins. This results in $D_U$ datasets with a size (in terms of # of columns and # of rows): (12, 3732), (27, 1178), (13, 18249) and (20, 140700), for tasks $T_1$ to $T_4$, respectively. Specifically, we applied $k$-means clustering over the active domain of each attribute (with a maximum $k$ set as 30), and derived equality literals, one for each cluster. We then compressed the input tables by replacing rows into tuple clusters, reducing the number of rows. This pragmatically help us avoid starting from large $D_U$ by only retaining the values of interests, and still yield desired skyline datasets. For $T_5$, a large bipartite graph is constructed with a size of (7925, 34) (# of edges, # of nodes' features). The generation of graphs consistently aligns with its table data counterpart, by conveniently replacing augment and reduction to their graph counterpart that performs link insertions and deletions.

**Evaluation metrics**. We adopt the following metrics to quantify the effectiveness of data discovery approaches. Denote $D_M$ as an initial dataset, and $\mathcal{D}_o$ a set of output datasets from a data discovery algorithm. (1) We define the *relative improvement* $\mathrm{RImp}(p)$ for a given measure $p$ achieved by a method as $\frac{M(D_M).p}{M(D_o).p}$. As all metrics are normalized to be minimized, the larger $\mathrm{RImp}(p)$ is, the better $D_p$ is in improving $M$ *w.r.t.* $p$. Here $M(D_M).p$ and $M(D_p).p$ are obtained by actual model inference test. This allows us to fairly compare all methods in terms of the quality of data suggestion. For efficiency, we compare the time cost of data discovery upon receiving a given model or task as a "query".

**Exp-1: Effectiveness**. We first evaluate MODis methods over five tasks. Results for $T_1$ and $T_3$ are shown in Fig. 7 (the outer, the better). While results for $T_2$ and $T_4$ are presented in Table 4. Results for $T_5$ are in Table 5. We also report the model performance over the input tables as a "yardstick" ("Original") for all methods. As all baselines output a single table, to compare MODis algorithms, we select the table in the Skyline set with the best estimated $p_{Acc}$, $P_{F1}$, $P_{MSE}$, $p_{Acc}$ and $p_{Pc_5}$ for $T_1$ to $T_5$, respectively. As METAM optimizes a single utility score, we choose the same measure for each task as the utility. We apply model inference to all the output tables to report actual performance values. We have the following observations.

(1) MODis algorithms outperform all the baselines in all tasks. As shown in Table 4, for example, for $T_4$, the datasets that bear best $p_{Acc}$ and the second best are returned by ApxMODis (0.9535) and BiMODis (0.9525), respectively, and all MODis methods generated datasets that achieve 0.87 on $p_{F1}$ in $T_2$.

(2) Over the same dataset and for other measures, MODis algorithms outperform the baselines in most cases. For example, in $T_1$, the result datasets that most optimize $p_{Fsc}$ and $p_{MI}$ are obtained by BiMODis and ApxMODis, respectively; also in $T_2$ and $T_3$, NOBiMODis and BiMODis show absolute dominance in most measures. Table 5 also verifies that MODis easily generalizes to suggest graph data for GNN-based analytics, beyond tabular data.

(3) Methods with data augmentation (*e.g.*,METAM and Starmie) enriches data to improve model accuracy, at a cost of training time, while feature selection methods (*e.g.*,SkSFM and H2O) reduce data at the cost of accuracy with improved training efficiency. MODis methods are able to balance these trade-offs better by *explicitly* performing multi-objective optimization. For example, $p_{Acc}$ and $p_{Train}$ in $T_4$, The best result for training cost (0.2359s) is contributed from SkSFM, yet at a cost of lowest model accuracy (0.8839).

We also compared $p_{Acc}$ on $T_4$ with HydraGAN, a generative data augmentation method, which achieves 0.9355 with 330 rows

| $T_2$: House | Original | METAM | METAM-MO | Starmie | SkSFM | H2O | ApxMODis | NOBiMODis | BiMODis | DivMODis |
|---|---|---|---|---|---|---|---|---|---|---|
| $p_{F1}$ | 0.8288 | 0.8510 | 0.8310 | 0.8351 | 0.7825 | 0.8333 | 0.9044 | **0.9125** | **0.9125** | 0.8732 |
| $p_{Acc}$ | 0.8305 | 0.8322 | 0.8333 | 0.8331 | 0.7826 | 0.8305 | 0.9050 | **0.9121** | **0.9121** | 0.8729 |
| $p_{Train}$ | 0.2000 | 0.21 | 0.19 | 0.2100 | 0.2000 | 0.2000 | 0.1533 | **0.1519** | **0.1519** | 0.2128 |
| $p_{F_{sc}}$ | 0.0928 | 0.0889 | 0.0894 | 0.0149 | 0.2472 | 0.0691 | 0.2268 | **0.2610** | **0.2610** | 0.2223 |
| $p_{MI}$ | 0.126 | 0.1109 | 0.1207 | 0.0243 | *0.2970* | 0.1054 | 0.2039 | 0.2018 | 0.2018 | **0.3164** |
| Output Size | (1178, 27) | (1178, 28) | (1178, 28) | (1178, 32) | (1178, 4) | (1178, 15) | (835, 17) | (797, 17) | (797, 17) | (1129, 5) |
| $T_4$: Mental | Original | METAM | METAM-MO | Starmie | SkSFM | H2O | ApxMODis | NOBiMODis | BiMODis | DivMODis |
| $p_{Acc}$ | 0.9222 | 0.9468 | 0.9462 | 0.9505 | 0.8839 | 0.9236 | **0.9532** | 0.9471 | 0.9525 | 0.9471 |
| $p_{Pc}$ | 0.7940 | 0.7991 | 0.8070 | 0.8106 | 0.6577 | 0.7892 | **0.8577** | 0.8454 | 0.8549 | 0.8454 |
| $p_{Rc}$ | 0.7722 | 0.7846 | 0.7959 | 0.8030 | 0.7523 | 0.7879 | **0.8097** | 0.8092 | 0.8075 | 0.8092 |
| $p_{F1}$ | 0.7829 | 0.7918 | 0.8014 | 0.8068 | 0.7018 | 0.7885 | **0.8330** | 0.8269 | 0.8305 | 0.8269 |
| $p_{AUC}$ | 0.9618 | 0.9757 | 0.9774 | 0.9784 | 0.9326 | 0.9615 | **0.9792** | 0.9755 | 0.9789 | 0.9755 |
| $p_{Train}$ | 0.4098 | 0.3198 | 0.4027 | 0.3333 | **0.2359** | 0.2530 | 0.3327 | 0.2818 | 0.3201 | 0.2818 |
| Output Size | ($10^5$, 14) | ($10^5$, 15) | ($10^5$, 15) | ($10^5$, 16) | ($10^5$ 8) | ($10^5$, 8) | (128332, 16) | (116048, 16) | (128332, 17) | (116048, 16) |

**Table 4: Comparison of Data Discovery Algorithms in Multi-Objective Setting ($T_2$, $T_4$)**

| $T_5$: Model | Original | ApxMODis | NOMODis | BiMODis | DivMODis |
|---|---|---|---|---|---|
| $p_{Pc_5}$ | 0.7200 | **0.8200** | 0.8000 | **0.8200** | 0.8000 |
| $p_{Pc_{10}}$ | 0.6600 | 0.8100 | 0.8000 | **0.8200** | 0.8000 |
| $p_{Rc_5}$ | 0.1863 | **0.2072** | 0.2022 | **0.2072** | 0.2022 |
| $p_{Rc_{10}}$ | 0.3217 | 0.3866 | 0.3816 | **0.3977** | 0.3816 |
| $p_{Nc_5}$ | 0.6923 | **0.7935** | 0.7875 | 0.7924 | 0.7875 |
| $p_{Nc_{10}}$ | 0.6646 | 0.7976 | 0.7891 | **0.8033** | 0.7891 |
| Output Size | (7925, 0) | (5826, 30) | (1966, 6) | (2869, 4) | (1966, 6) |

**Table 5: Comparison of MODis Methods on $T_5$**

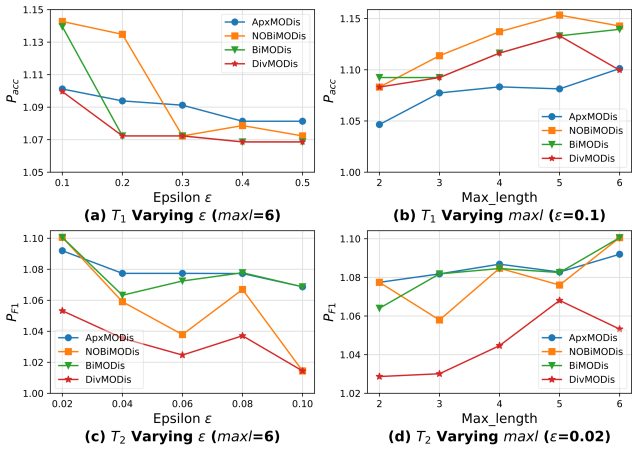



**Figure 8: Effectiveness: Impact of Factors**

but fell short of data discovery methods. Increasing the number of rows further reduced performance, reflecting the limitations of generative approaches in this context, which cannot utilize verified external data sources, and synthetic data often lacks inherent reliability and contextual relevance of discovered data.

**Exp-2: Impact factors**. We next investigate the MODis methods under the impact of two factors: $\epsilon$ and the maximum path length (maxl), as well as the impact of $\alpha$ on DivMODis.

*Varying* $\epsilon$. Fixing maxl = 6, we varied $\epsilon$ from 0.5 to 0.1 for $T_1$. As shown in Fig. 8(a), MODis algorithms are able to improve the model in $p_{acc}$ better with smaller $\epsilon$, as they all ensure to output a $\epsilon$-Skyline set that better approximate a Skyline set when $\epsilon$ is set to be smaller. In all cases, they achieve a relative improvement rImp($p_{Acc}$) at least 1.07. BiMODis and NOBiMODis perform better in recognizing better solutions from both ends in reduction and augmentation as smaller $\epsilon$ is enforced. ApxMODis, with reduction only, is less sensitive to the change of $\epsilon$ due to that larger $\epsilon$ may "trap" it to local optimal sets from one end. Adding diversification (DivMODis) is able to strike a balance between ApxMODis and BiMODis by enforcing to choose difference datasets out of local optimal sets, thus improving ApxMODis for smaller $\epsilon$. We choose a smaller range of $\epsilon$ for $T_2$ in Fig. 8(c), as the variance of $p_{F1}$ is small. As $\epsilon$ varies from 0.1 to 0.02, NOBiMODis improves F1 score from 0.84 to 0.91.

*Varying* maxl. Fixing $\epsilon$ = 0.1, we varied maxl from 2 to 6. Fig. 8(b, d) tells us that all MODis algorithms improve the task performance for more rounds of processing. Specifically, BiMODis and NOBiMODis benefit most as bi-directional search allows both to find better solution from wider search space as maxl becomes larger. ApxMODis is less sensitive, as the reduction strategy from dense tables incurs smaller loss in accuracy. DivMODis finds datasets that ensure best model accuracy when maxl = 5, yet may "lose chance" to maintain the accuracy, due to that the diversification step may update the Skyline set with less optimal but more different counterparts in future levels (*e.g.*, when maxl = 6).

*Varying* $\alpha$ *in* DivMODis. We demonstrate the effectiveness of DivMODis by adjusting $\alpha$. A smaller $\alpha$ prioritizes performance, while a larger $\alpha$ emphasizes content diversity, measured by hamming distance. Fig. 9(a) illustrates *Performance Diversity*, where smaller $\alpha$ results in a wider accuracy range with a balanced and stable distribution. Both the mean and median remain centered. As $\alpha$ increases, the accuracy distribution narrows and shifts toward higher values, reflecting the dominance of high-accuracy datasets in the Skyline set. Fig. 9(b) verifies the impact of *Content Diversity*, visualized as the percentage contribution of each adom. Larger $\alpha$ leads to more evenly distributed contributions. The standard deviation values above the heatmap quantify this trend, showing a consistent decrease as $\alpha$ increases, indicating improved balance.

**Exp-3: Efficiency and Scalibility**. We next report the the efficiency of MODis algorithms for task $T_1$ and $T_3$ over Kaggle and HF, respectively, and the impact of factors $\epsilon$ and maxl. We also evaluate their scalability for $T_1$ and $T_5$ in terms of input size.

*Efficiency: Varying* $\epsilon$. Fixing maxl = 6 and varying $\epsilon$ from 0.1 to 0.5, Fig. 10 (a) verifies the following. (1) BiMODis, NOBiMODis and DivMODis take less time as $\epsilon$ increases, as a larger $\epsilon$ provides more chance to prune unnecessary valuations. DivMODis

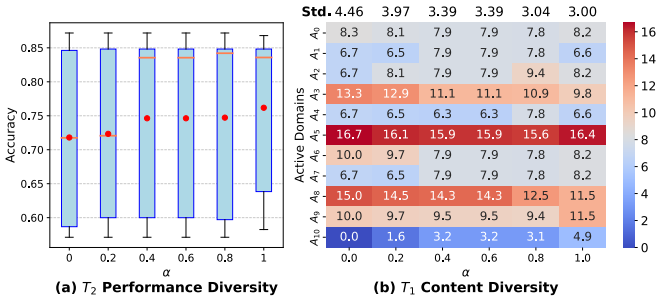



**Figure 9: Impact of $\alpha$ for DivMODis**

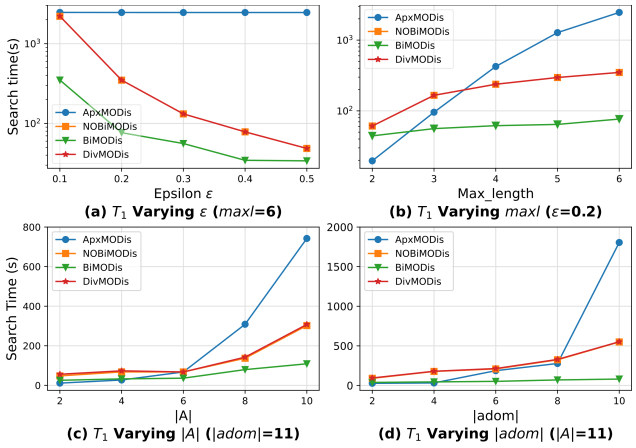



**Figure 10: Efficiency and Scalabilitiy**

has a comparable performance with NOBiMODis, as it mainly benefits from the bi-directional strategy, which exploits early pruning and a stream-style placement strategy. (2) As shown in Fig. 10(a), for $T_1$, BiMODis, NOBiMODis, and DivMODis are 2.5, 2, and 2 times faster than ApxMODis on average, respectively. ApxMODis takes longer time to explore a larger universal table with reduct operators. It is insensitive to $\epsilon$. We observe that its search from the "data rich" end may converge faster at high-quality $\epsilon$-Skyline sets.

*Efficiency: Varying* maxl. Fixing $\epsilon = 0.2$ for task $T_1$ and $\epsilon = 0.1$ for task $T_3$, we varied maxl from 2 to 6, all MODis algorithms take longer as maxl increases, as shown in Fig. 10 (b). Indeed, larger maxl results in more states to be valuated, and more non-$\epsilon$-dominance relation to be resolved. ApxMODis is sensitive to maxl due to the rapid growth of the search space. In contrast, BiMODis mitigates the impact with bi-directional strategy and effective pruning.

*Scalability*. We varied the number of total attributes $|A|$ and size of the largest active domain |adom|. We perform $k$-means clustering over the tuples of the universal table with $k = |\text{adom}|$, and extended operators with range queries to control |adom|. Fig. 10 (c) and (d) show that all MODis algorithms take more time for larger $|A|$ and |adom|. BiMODis scales best due to bi-directional strategy. DivMODis remains more efficient than ApxMODis, indicating affordable overhead from diversification.

While our algorithms scale well with $|A|$ and |adom|, high-dimensional datasets may present challenges due to the search space growth. Dimensionality reduction such as PCA or feature selection, or correlation-based pruning (to identify and eliminate highly correlated or redundant features), can be tailored to specific tasks to mitigate these challenges.

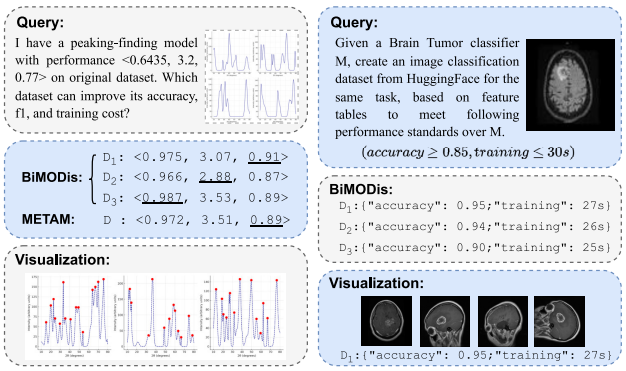



**Figure 11: Case 1 (left): Discover Datasets for Materials Peak Classification Analysis. Case 2 (right): Test Data Generation for Model Performance Benchmarking**

**Exp-4: Case study**. We next report two real-world case studies to illustrate the application scenarios of MODis.

*(1) "Find data with models".* A material science team trained a random forest-based classifier to identify peaks in 2D X-ray diffraction data. They seek more datasets to improve the model's *accuracy, training cost, and F1 score* for downstream fine-tuning. Original X-ray datasets and models are uploaded to a crowd-sourced X-ray data platform we deployed [43] with best performance of $< 0.6435, 3.2, 0.77 >$. Within available X-ray datasets, BiMODis created three datasets $\{D_1, D_2, D_3\}$ and achieved the best performance of 0.987, 2.88, and 0.91, respectively. We set METAM to optimize F1-score, and achieved a performance score of $< 0.972, 3.51, 0.89 >$ over its output dataset. Fig. 11 illustrates such a case that is manually validated with ground-truth from a third-party institution.

*(2) Generating test data for model evaluation.* We configure MODis algorithms to generate test datasets for model benchmarking, where specific performance criteria can be posed [40]. Utilizing a trained scientific image classifier from Kaggle, and a pool of image feature datasets $\mathcal{D}$ from HF with 75 tables, 768 columns, and over 1000 rows. We request BiMODis to generate datasets over which the classifier demonstrates: "accuracy > 0.85" and "training cost < 30s." BiMODis successfully generated 3 datasets to be chosen from within 15 seconds, with performance $< 0.95, 0.27 >$, $< 0.94, 0.26 >$ and $< 0.90, 0.25 >$, as in Fig. 11.

## 7 CONCLUSION

We have introduced MODis, a framework that generate skyline datasets to improve data science models on multiple performance measures. We have formalized skyline data generation with transducers equipped with augment and reduction operators. We show the hardness and fixed-parameter tractability of the problem. We have introduced three algorithms that compute approximate Skyline sets in terms of $\epsilon$-Skyline set, with reduce-from-universal, bi-directional, and diversification paradigms. Our experiments have verified their effectiveness and efficiency. A future topic is to enhance MODis with query optimization techniques to scale it for larger input with high-dimensional data. Another topic is to extend MODis for distributed Skyline data generation.

## ACKNOWLEDGMENTS

This work is supported by NSF under OAC-2104007.

## REFERENCES


[1] U.S. General Services Administration. 2023. Data.gov. https://www.data.gov/

[2] Amit Chakrabarti and Sagar Kale. 2015. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming* (2015).

[3] Jan Chomicki, Paolo Ciaccia, and Niccolo' Meneghetti. 2013. Skyline queries, front and back. *ACM SIGMOD Record* 42, 3 (2013), 6–18.

[4] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.

[5] Chance DeSmet and Diane Cook. 2024. HydraGAN: A Cooperative Agent Model for Multi-Objective Data Generation. *ACM Transactions on Intelligent Systems and Technology* 15, 3 (2024), 1–21.

[6] AnHai Doan, Alon Halevy, and Zachary Ives. 2012. *Principles of data integration.*

[7] Bin Dong, Kesheng Wu, Surendra Byna, Jialin Liu, Weijie Zhao, and Florin Rusu. 2017. ArrayUDF: User-defined scientific data analysis on arrays. In *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*. 53–64.

[8] Zahra Donyavi and Shahrokh Asadi. 2020. Diverse training dataset generation based on a multi-objective optimization for semi-supervised classification. *Pattern Recognition* 108 (2020), 107543.

[9] Iddo Drori, Yamuna Krishnamurthy, Raoni Lourenco, Remi Rampin, Kyunghyun Cho, Claudio Silva, and Juliana Freire. 2019. Automatic machine learning by pipeline synthesis using model-based reinforcement learning and a grammar. *arXiv preprint arXiv:1905.10345* (2019).

[10] Mahdi Esmailoghli, Christoph Schnell, Renée J Miller, and Ziawasch Abedjan. 2023. Blend: A unified data discovery system. *arXiv preprint arXiv:2310.02656* (2023).

[11] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée J Miller. 2023. Semantics-Aware Dataset Discovery from Data Lakes with Contextualized Column-Based Representation Learning. *Proceedings of the VLDB Endowment* 16, 7 (2023).

[12] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2018. Practical automated machine learning for the automl challenge 2018. In *International Workshop on Automatic Machine Learning at ICML*. 1189–1232.

[13] Sainyam Galhotra, Yue Gong, and Raul Castro Fernandez. 2023. Metam: Goal-oriented data discovery. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2780–2793.

[14] H2O.ai. 2022. *H2O: Scalable Machine Learning Platform.* https://github.com/h2oai/h2o-3 version 3.42.0.2.

[15] Pierre Hansen. 1980. Bicriterion path problems. In *Multiple Criteria Decision Making Theory and Application: Proceedings of the Third Conference Hagen/Königswinter, West Germany, August 20–24, 1979*. 109–127.

[16] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[17] Zezhou Huang, Pranav Subramaniam, Raul Castro Fernandez, and Eugene Wu. 2023. Kitana: Efficient Data Augmentation Search for AutoML. *arXiv preprint arXiv:2305.10419* (2023).

[18] Hugging Face AI 2023. Hugging Face – The AI Community Building the Future. https://huggingface.co/

[19] John T Hwang and Joaquim RRA Martins. 2018. A fast-prediction surrogate model for large datasets. *Aerospace Science and Technology* 75 (2018), 74–87.

[20] Kaggle. 2023. Kaggle: Your Home for Data Science. https://www.kaggle.com/

[21] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).

[22] Mina Konakovic Lukovic, Yunsheng Tian, and Wojciech Matusik. 2020. Diversity-guided multi-objective bayesian optimization with batch evaluations. *Advances in Neural Information Processing Systems* 33 (2020), 17708–17720.

[23] Hsiang-Tsung Kung, Fabrizio Luccio, and Franco P Preparata. 1975. On finding the maxima of a set of vectors. *J. ACM* 22, 4 (1975), 469–476.

[24] Berti-Equille Laure, Bonifati Angela, and Milo Tova. 2018. Machine learning to data management: A round trip. In *ICDE*. 1735–1738.

[25] Maurizio Lenzerini. 2002. Data integration: A theoretical perspective. In *PODS*.

[26] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. 2017. Feature selection: A data perspective. *ACM computing surveys (CSUR)* 50, 6 (2017), 1–45.

[27] Yuliang Li, Xiaolan Wang, Zhengjie Miao, and Wang-Chiew Tan. 2021. Data augmentation for ml-driven data preparation and integration. *Proceedings of the VLDB Endowment* 14, 12 (2021), 3182–3185.

[28] Chunming Liu, Xin Xu, and Dewen Hu. 2014. Multiobjective reinforcement learning: A comprehensive overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 3 (2014), 385–398.

[29] Andre KY Low, Flore Mekki-Berrada, Aleksandr Ostudin, Jiaxun Xie, Eleonore Vissol-Gaudin, Yee-Fun Lim, Abhishek Gupta, Qianxiao Li, Yew Soon Ong, Saif A Khan, et al. 2023. Evolution-guided Bayesian optimization for constrained multi-objective optimization in self-driving labs. (2023).

[30] Tien-Dung Nguyen, Tomasz Maszczyk, Katarzyna Musial, Marc-André Zöller, and Bogdan Gabrys. 2020. Avatar-machine learning pipeline evaluation using surrogate model. In *Advances in Intelligent Data Analysis XVIII: 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27–29, 2020, Proceedings 18*. Springer, 352–365.

[31] Xuan Vinh Nguyen, Jeffrey Chan, Simone Romano, and James Bailey. 2014. Effective global approaches for mutual information based feature selection. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 512–521.

[32] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D Lawrence. 2022. Challenges in deploying machine learning: a survey of case studies. *Comput. Surveys* 55, 6 (2022), 1–29.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[34] Hanchuan Peng, Fuhui Long, and Chris Ding. 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence* 27, 8 (2005), 1226–1238.

[35] Yuji Roh, Geon Heo, and Steven Euijong Whang. 2019. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering* 33, 4 (2019), 1328–1347.

[36] Paolo Serafini. 1987. Some considerations about computational complexity for multi objective combinatorial problems. In *Recent Advances and Historical Development of Vector Optimization: Proceedings of an International Conference on Vector Optimization Held at the Technical University of Darmstadt, FRG, August 4–7, 1986*. Springer, 222–232.

[37] Darius Šidlauskas and Christian S Jensen. 2014. Spatial joins in main memory: Implementation matters! *Proceedings of the VLDB Endowment* 8, 1 (2014), 97–100.

[38] Tom F Sterkenburg and Peter D Grünwald. 2021. The no-free-lunch theorems of supervised learning. *Synthese* 199, 3-4 (2021), 9979–10015.

[39] George Tsaggouris and Christos Zaroliagis. 2009. Multiobjective optimization: Improved FPTAS for shortest paths and non-linear objectives with applications. *Theory of Computing Systems* 45, 1 (2009), 162–186.

[40] Francesco Ventura, Zoi Kaoudi, Jorge Arnulfo Quiané-Ruiz, and Volker Markl. 2021. Expand your training limits! generating training data for ml-based data management. In *SIGMOD*.

[41] Mengying Wang, Sheng Guan, Hanchao Ma, Yiyang Bian, Haolai Che, Abhishek Daundkar, Alp Sehirlioglu, and Yinghui Wu. 2023. Selecting Top-k Data Science Models by Example Dataset. In *CIKM*.

[42] Mengying Wang, Hanchao Ma, Yiyang Bian, Yangxin Fan, and Yinghui Wu. 2025. Generating Skyline Datasets for Data Science Models. arXiv:cs.DB/2502.11262 https://arxiv.org/abs/2502.11262

[43] Mengying Wang, Hanchao Ma, Abhishek Daundkar, Sheng Guan, Yiyang Bian, Alpi Sehirlioglu, and Yinghui Wu. 2022. CRUX: crowdsourced materials science resource and workflow exploration. In *CIKM*.

[44] Chengrun Yang, Jicong Fan, Ziyang Wu, and Madeleine Udell. 2020. Automl pipeline selection: Efficiently navigating the combinatorial space. In *proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 1446–1456.

[45] Zhuoyue Zhao, Feifei Li, and Yuxi Liu. 2020. Efficient join synopsis maintenance for data warehouse. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2027–2042.

[46] JH Zheng, YN Kou, ZX Jing, and QH Wu. 2019. Towards many-objective optimization: Objective analysis, multi-objective optimization and decision-making. *IEEE Access* 7 (2019), 93742–93751.

[47] Patrick Ziegler and Klaus R Dittrich. 2007. Data integration—problems, approaches, and perspectives. In *Conceptual modelling in information systems engineering*. Springer, 39–58.