Multi-Objective Data Discovery for Data Science Models

ABSTRACT

Data discovery has been studied to identify relevant data to boost the performance of data science models in data-intensive tasks. In practice, there often comes the need to collect the data that can simultaneously optimize or trade among multiple performance metrics (such as training cost, accuracy, or inference cost) for specific models. Existing data augmentation techniques that apply table-wise data integration or feature selection are constrained to a single utility goal and meanwhile not sufficiently fine-grained to scale to multiple performance metrics. This paper introduces a novel paradigm to automatically create data from multiple sources with a goal to improve expected performance of a specific data science model in terms of multiple user-defined performance metrics. We formulate the problem as a multi-objective data discovery, and establish a formal computation system in terms of finite state transducer with finer grained operators enhanced by selection conditions. We analyze its expressiveness, and establish the hardness and approximability results. We then provide feasible algorithms that can create and maintain multiple datasets, from approximation scheme to fast heuristics that maintains the solution, all with guarantees in terms of Pareto optimality. We experimentally verify the efficiency and effectiveness of our algorithms in data creation. We also showcase of their applications in accelerating the design of scientific data analytical pipelines.

1 INTRODUCTION

Data-driven analytical pipelines with data scienc models are routinely processed in a wide range of applications. Such pipelines rely on high quality data science (machine learning) models. Among the challenges is the effective selection and creation of datasets that lead to high-quality models. In other words, *how to create new data to improve the overall (expected) performance of a model?*

Given a data science model M and a set of tables $\mathcal{D} = \{D_1, \ldots D_n\}$, data discovery aims to properly select *and* refine the data from $\mathcal{D}$ to obtain a new data table D', such that the expected performance of M over D' is improved. Moreover, the model M is evaluated by *multiple* (user-specified) performance measures such as training cost, processing cost, accuracy, inference cost, memory consumption, etc. It is desirable to suggest a dataset, over which the model has desirable performances in terms of *all* the measures.

Crowdsourced data platforms such as HuggingFace [10] provide portals to make datasets and models available. Data augmentation [26] and feature selection [18] have been also separately studied to improve machine learning, by carefully choosing useful data sources and feature space, respectively. Data integration has been adopted as an enabling technique to create new data for data augmentation [3]. Nevertheless, these approaches are not optimized where data science models are the first-class citizen. There still lacks effective solutions that (1) can suggest data that directly response to a model as a "query" (*i.e.,* "search data with a model"); and (2) in particular, improve the expected performances of the model in the presence of *multiple* performance measures.

We consider the following real-world example.

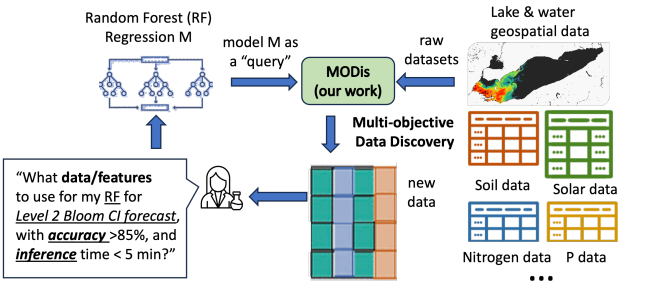

Figure 1: Discover and integrate features from multiple data sources improves the performance of a trained random forest (RF) for predicting CI index in both accuracy and training cost, as required by users. Manual tuning with cross-validation and feature selection tools finds 10 features from more than 50 after several weeks. Our framework automatically suggested a new dataset with 8 features with comparable model accuracy and training cost, in a few minutes.

Example 1.1. To understand the impact of harmful algal bloom (HABs) of a lake and its reason, a research team need a model that is able to forecast the chlorophyll-a index (CI-index), a critical algal bloom quantification index, in a short term. The team has collected a set of tables that involve more than 50 factors (*e.g.,* fertilizer, water quality, weather) involving several upstream rivers and watershed systems of the lake. The team has already initialized a random forest (RF) model from small scale tests, and wish to (1) improve its performance in terms of *multiple* metrics, including two accuracy measures: root mean square error (*RMSE*), and R^2 test, for "Level 2 bloom" CI-index, and its training cost; (2) understand *what* are new important factors and their values to be considered to achieve such improvement; and (3) track *where* the values are from and *how* they are integrated from original data.

A typical process involves an excessive number of trials that repeats a pipeline of (manual) hyper-parameter tuning, feature selection, data integration, cross validation, and inference tests. It turns out that only 8 key features with certain ranges of values are needed to achieve a regression with desirable accuracy at 89.6% (quantified by RMSE and R^2 test). This process took the team 8 weeks to get the desired model, and the following observations.

- (1) not all water data tables from every upstream watershed systems are needed, but only two of them – one from a basin area and one upstream river;
- (2) A single physicochemical attribute "`Phosphorus`" alone does not determine C1-index well for recent years' CI-index, but "`Nitrogen`" and "`Phosphorus`" both jointly determines CI-index, as an emerging influence pattern; and
- (3) The original data contains, for both columns "`Nitrogen`" and "`Phosphorus`", the values over the past 11 years. However, for an accurate short-term prediction in 2023, the fraction of the seasonal "*Spring*" data and in particular years "*2013*" and "*2015*" are important, due to high nutrition loads.

The above example calls for an automated data discovery process that can (1) suggest data from available ones to improve data science models for *multiple measures* at scale, and (2) provide an integrative solution to suggest "what" data to choose and "how" to integrate the selected data. Moreover, the overall process should avoid expensive retraining and inference cost of the model.

One may exploit established data integration approaches to identify "joinable" tables and test the model. This requires an exhaustive enumeration of all the plans, leaving alone the overhead for actual evaluation. Such evaluation may already be expensive especially when M has high model complexity or with expensive user defined functions [19], or involves nontrivial in-lab experiments or simulations as seen in scientific workflows [23]. Goal-driven data integration provides guided design for this process, yet may only response to optimizing a single performance metric rather than multi-objective improvement for specific models.

Example 1.2. The research team may issue an analytical query in the following form: "*What are proper dataset for which our random forest model can have an improved expected accuracy for forecasting 'Level 2 bloom' CI-index, with accuracy in terms of RMSE at most* 0.3, R^2 *score at least* 0.7, *and training cost no more than 5 minutes?*"

At first glance this seems "contradicting" due to the trade-off between training cost and model accuracy for the two measures on CI-index. Nevertheless, a data discovery process can still compromise to suggest one or more datasets, over which the model is able to "show case" of desired accuracy for at least one metric (*e.g.,* a $\geq$ 86% accuracy for forecasting Level 2 bloom) and demonstrate comparable performance to the rest two metrics. This encourages us to seek for data discovery that can (1) combine data integration and multi-objective model evaluation, (2) pursue *multi-objective* optimization. Both are *new* challenges that are not well addressed by existing data discovery methods for improving data-driven pipelines.

Moreover, the computation should not be limited to data augmentation by joining columns, but also to explore proper *selection conditions* and prune irrelevant rows. This reduces unnecessary overhead for downstream fine-tuning effort of the model.

This paper proposes a novel framework, from a formal characterization of the data discovery process to feasible algorithms with provable guarantees, to address the above two challenges. Our main technical contributions are as follows.

(1) In response to the need of suggesting data with both columns and rows, We introduce a formal characterization of data discovery computation in terms of a *finite state transducer*, equipped with fine-grained operators with selection conditions (Section 3). We study the generality and property of the system and verify its expressiveness. We show that the system resembles of data integration and feature selection. Better still, it has a simple form that ensures non-blocking operations.

(2) Based on our formulation of data discovery system, we address the second challenge and introduce a novel problem of multi-objective data discovery problem (MODis) (Section 4). We model the problem as a multi-objective optimization process of the running of the data discovery system, which performs sequential operators to update an initial dataset. The running aims to outputs a *Pareto set* of datasets, over which a model is expected to have optimized performance over multiple evaluation metrics.

(3) While the problem is in general intractable, we show that there exists a fully polynomial time approximation scheme (FPTAS) for a constrained version of the problem (Section 5) under a fixed parameter specification. Our algorithm adopts a performance-driven search process, which interleaves multiple rounds of data integration and cost-effective model evaluation over historically observed tests and their performances.

Moreover, we introduce two variants of the algorithm to address the practical needs. The first exploits known correlations of the performance metrics to perform early termination and pruning, via a bi-directional search scheme (Section 5.3). The second introduce a diversification process to improve the generality of the generated datasets (Section 5.4). Both preserve the quality guarantee in terms of Pareto optimality.

(5) We finally introduce an efficient maintenance algorithm to maintain the datasets upon the updates of the underlying source dataset.

Using real benchmark datasets, models and analytical tasks, we experimentally verify the effectiveness of our data discovery scheme in improving the model performance and the efficiency of data generation. For example, revise as needed our algorithms take **TBF** time to generate new data to improve pre-trained **TBF** (regression, classification) models by **TBF** in accuracy and **TBF** in training cost. It outperforms other approaches that seperately performs data augmentation or feature selection alone; and is feasible for large dataset. Our case study verifies its application in suggesting data for other applications such as transfer learning or benchmarking.

Related works. We categorize related works as follows. We remark that these work involve a common goal on improving the performance of data analysis models.

Feature Selection. Feature selection is a cornerstone task to remove irrelevant and redundant attributes and identify important ones for model training. Common strategies include filtering, wrapping, and embedding [18]. Filtering methods rank features in terms of correlation or mutual information [22, 25] 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. Wrappers [13] choose features based on model performance estimated by a predictive model. The search cost is often extensive. Moreover, they risk overfitting due to optimizing a single criteria of model accuracy. Embedded methods directly learn feature selection models such as Lasso Regression, by penalizing (layer-wise) feature weights to improve model accuracy [31]. While they can be more efficient than wrapping, their performance are more sensitive to model architectures and task types, hyper-parameter tuning and additional domain-specific knowledge from dataset and models.

Our method differs in that (1) it creates new data with both data augmentation and masking of less useful values, rather than simply dropping entire feature columns; (2) it finds data that improves model in terms of multiple performance measures, beyond a single (accuracy) measure; and (3) the computation does not require internal knowledge of the models but simply data integration and masking operators, without incurring learning overhead. The

process resemble SPJ (select, project, join) operations and readily benefit from well established query optimization techniques.

Data Augmentation. Data augmentation and integration aims to create data from multiple data sources towards a unified view [3, 33]. It is often specified to improve data completeness and richness [26] and may be sensitive to the quality of schema that in turn require entity matching [17] and schema matching [12]. Our method has a goal to create dataset to improve expected performance of given data-driven models and analytical pipelines. This is different from the goal of conventional data integration that mostly focus on improving the completeness of data and query answers alone.

Closer to our work is goal-driven data discovery [5, 6]. The methods perform data augmentation queries (joins) to enrich input data to improve the performance of given tasks. The process discovers and augments datasets as path plans, and achieves a desired performance quality with a bounded number of queries. Our approach differs from these work as follows. (1) We formalize data discovery with a more expressive model with finer grained column augmentation and cell-level operators with selection conditions. This allows us to create data with more expressiveness, and to better mitigate over-fitting or under-fitting issues. (2) We target at multi-objective goals beyond a single performance measure. We provide algorithms with quality guarantees in terms of Pareto optimality, as well as optimization techniques. These are not addressed in prior methods.

Crowdsourced Data Services. Several platforms are available to allow users to share and search datasets, such as Dataset search [2], Data.gov [1], Kaggle [11], Hugging Face [10], and Zenodo [4]. These services exploit user-defined tags to retrieve relevant datasets, yet lack the necessary capability to provide datasets towards improving the expected performance for specific models or tasks. Our approach is among the first effort with the enhanced capability to directly create datasets that improve the expected performance of models as "queries", and in terms of multiple performance measures.

2 MODELS AND PERFORMANCE EVALUATION

Datasets. A dataset $D(A_1, \ldots A_m)$ is a structured table instance that conform to a relational 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 ith attribute A_i of t is assigned a value a. A dataset may have missing values at some attribute A (*i.e.,* $t.A = \emptyset$).

We denote as $\mathcal{D} = \{D_1, \ldots D_n\}$ a set of given datasets. Each dataset D_i confirms to a schema R_i. We denote as $\mathcal{A}$ the set of all the attributes from the datasets in $\mathcal{D}$. Given an attribute $A \in \mathcal{A}$, the *active domain* of A, denoted as $\text{adom}(A)$, refers to the finite set of all the values of A occurred in $\mathcal{D}$.

Models. We characterize a data science model as a function in the general form of $M : D \to \mathbb{R}^d$, 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 set. 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.

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 data science models for the need of consistency and accuracy required in data science pipelines.

Symbol	Notation
$\mathcal{D}$, D, R_D	a set of datasets, dataset, and schema
$\mathcal{A}$, A, $\text{adom}(A)$	attribute set, attribute, and active domain
M	a data science model $D \to \mathbb{R}^d$
$\mathcal{P}$, p, (p_l, p_u)	performance measures, a measure, and its range
T, $t = (M, D, \mathcal{P})$, $t.\mathcal{P}$	a test set; a single test, and 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
$C = (s_M, M, T, \mathcal{E})$	a configuration of data discovery system $\mathcal{T}$
$G_{\mathcal{T}} = (\mathcal{S}, \delta)$	transition graph

Table 1: Table of notations needs to be updated

Model Evaluation. Given a model M and an input dataset D, a *test* t is a triple $(M, D, \mathcal{P})$ that specifies a test dataset D, a pre-trained model M, and a set of (user defined) *performance measures* $\mathcal{P} = \{p_1, \ldots p_l\}$. A *measure* p is an indicator to be *maximized* such as the model accuracy *e.g.,* precision, recall, F1 (for classification); or minimized *e.g.,* mean average error for regression tasks, training time, inference time, or memory consumption.

A performance measure $p \in \mathcal{P}$ can be always verified by an actual application of M over D, yet this can be expensive, especially when it involves in-lab tests or human inspection. We assume the measure p can be efficiently *approximated* by an estimation model $\mathcal{E}$ (or simply "estimator") [9, 30]. An estimator $\mathcal{E}$ makes use of a set of historically observed tests of M with performance vectors (denoted as T) to infer the performance of M over new data D.

We use the following notations. (1) We unify $\mathcal{P}$ as a set of normalized measures to be *minimized*, within a range $(0, 1]$. For a measure to be maximized (*e.g.,* accuracy), one can easily convert it to an inversed counterpart (*e.g.,* relative error). (2) Each measure $p \in \mathcal{P}$ has a user-specified pair of lower and upper bounds p_l, p_u to be normalized in $(0, 1]$, *i.e.,* $0 < p_l \leq p_u \leq 1$. The range $[p_l, p_u]$ may specify tolerable training costs, memory consumption, or error ranges. (3) A test tuple $t = (M, D, \mathcal{P})$ can be *valuated* by an estimator $\mathcal{E}$, by assigning the value of each measure $p \in \mathcal{P}$ in polynomial time. The *performance vector* $t.\mathcal{P}$ is a valuated tuple where each entry has a constant $t.p$ of the corresponding metric $p \in \mathcal{P}$, estimated by $\mathcal{E}$.

Example 2.1. Consider Example 1.1. A pretrained random forest (RF) model M that predicts CI-index is evaluated by three measures $\mathcal{P} = \{\text{RMSE}, \text{R}^2, \text{T}_{\text{train}}\}$, 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 $(0, 0.6]$, R^2 in $[0, 0.35]$ *w.r.t.* a "inversed" lower bound $1 - 0.65$, and T_{train} in $(0, 0.5]$ *w.r.t.* an upper bound of "3600 seconds" (*i.e.,* no more than 1800 seconds)

An estimator $\mathcal{E}$ can be a regression model that learns from historical tuning records T to predict the performance of M given a new dataset D. In our study, we train an estimator $\mathcal{E}$ as a multi-output Gradient Boosting Model (MO-GBM) in scikit-learn library [24]. This type of regressor outputs predicted values for multiple variables, thus allows us to valuate the entire performance vector $t.\mathcal{P}$ for each test t with one call of $\mathcal{E}$.

We summarize the main notations in Table 1.

3 DATA DISCOVERY: A FORMALIZATION

Given datasets $\mathcal{D}$ and a fixed deterministic model M, we formalize a data discovery system in terms of finite state transducers. This formalization not only provides us the design principles of feasible data discovery algorithms, but also verifies necessary properties that lead to their provable optimality guarantees.

3.1 Data Discovery System

We characterize a *data discovery system* as a finite state transducer, denoted as $\mathcal{T} = (s_M, \mathcal{S}, \mathcal{O}, \mathcal{S}_F, \delta)$, where

- $\mathcal{S}$ is a set of states,
- $s_M \in \mathcal{S}$ is a designated start state,
- $\mathcal{O}$ is a set of operators of types $\{\oplus, \ominus\}$;
- $\mathcal{S}_F$ is a set of output states; and
- δ 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 adom_s. For each attribute $A \in R_s$, $\text{adom}_s(A) \subseteq \text{adom}(A)$ refers to a fraction of values A can take at state s. $\text{adom}_s(A)$ 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 $\text{adom}(A)$.

Operators. We consider two classes of "shorthanded" operators to generate new data from an input dataset. These operators can be expressed and processed as SPJ (select, project, join) queries, enhanced with data augmentation or feature selection semantics.

(1) *Augmentation*: denoted as $\oplus_c(D_M, D)$, which augments a dataset D_M with dataset $D \in \mathcal{D}$ subject to a selection condition c. Here c is a single literal in the form of $A = a$ (an equality condition).

An augmentation operator $\oplus_c(D_M, D)$ performs the following queries: (a) augment the schema R_M of D_M with an attribute A from the schema R of D, if $A \notin R_M$, and (b) integrate D_M with tuples from D with values of $R.A$ satisfy value constraints c; and (c) fill in the rest cells with "null" if there value of A is not known.

(2) *Reduction* $\ominus_c(D_M)$: mask the cells in D_M where the values of an attribute $R_M.A$ satisfy a literal constraint c with "null". Here c is a single literal in the form of $A = a$, with $a \in \text{adom}(A)$.

Transitions. Given a state s and a set of operators $\mathcal{O}$ of the two classes $\{\oplus, \ominus\}$, applying an operator $\text{op} \in \mathcal{O}$ over s creates a new *result* state s'. We represent this as a *transition*, denoted as $r = (s, op, s')$, where s' is the result of applying op to s.

Running. A *configuration* of $\mathcal{T}$, denoted as $C = (s_M, M, T, \mathcal{E})$, initializes a start state s_M (with a dataset D_M), a fixed deterministic model M, an estimator $\mathcal{E}$, and a test set 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.

- Starting from s_M, and at each state s, $\mathcal{T}$ iteratively applies operators from $\mathcal{O}$, whenever applicable, to either augment a state with new attributes and tuples, or drop tuples from the state. This spawns a set of children of a state.
- For each transition $r = (s, op, s')$ that is spawned with a result s', $\mathcal{T}$ (1) constructs a test tuple $t(M, s'.D, \mathcal{P})$, (2) invokes $\mathcal{E}$ *at runtime* to valuate t, and (3) adds t to T.

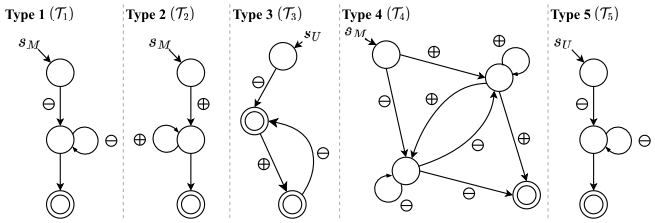

Figure 2: Data Discovery Systems: Specifications

- The above process *terminates* at a set of output states $\mathcal{S}_F$, under the validation of an external termination condition, or no new transition can be spawned.

The *result* of a running of $\mathcal{T}$ 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 $s.D_s$, for simplicity, we shall use a single general term "output", denoted as $\mathcal{D}_F$, to refer to output states or datasets as needed.

Example 3.1. Fig. 2 illustrates several classes of data discovery systems, categorized by the type of computation they perform.

(1) **Type 1** (resp. **Type 2**) system, denoted as $\mathcal{T}_1$ (resp. $\mathcal{T}_2$) starts from an initial dataset D_M in its start state s_M, and only applies reduction (resp. augmentation operators) to update D_M.

(2) **Type 3** system, denoted as $\mathcal{T}_3$, follows an "reduct-then-augment" transitions, which iteratively perform (a) a reduction operator to remove values from an initial dataset under selection conditions, followed by (b) an augmentation the reduced D_M with new attributes and their values. In particular, it may starts with a "universal" dataset D_u obtained by joining all attributes from a given set of datasets $\mathcal{D}$ (with a designated universal state, denoted as s_U).

(3) **Type 4** system, denoted as $\mathcal{T}_4$ is in a general form (that resembles Types 1-3 systems) that perform either augmentation or reduction at any step to create datasets.

(4) **Type 5** system is a variant of Type 1 under a configuration that starts from a universal dataset D_U as aforementioned. It adopts a "reduce-from-universal" strategy and only performs reduction operators to remove cell values from D_U.

Transition graph. A running of $\mathcal{T}$ can be naturally represented as the dynamic generation of a *transition graph* $G_{\mathcal{T}} = (\mathcal{S}, \delta)$, which is a directed acyclic graph (DAG) with a set of state nodes $\mathcal{S}$, 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 consecutive application of operators that converts an initial state s_1 with dataset D_1 to a new counterpart s_k with D_k. Consistently, we say a state node $s \in \mathcal{S}$ is *valuated*, if a corresponding test $t(M, s, D_s, \mathcal{P})$ is valuated by $\mathcal{E}$.

Example 3.2. Revisiting Example 1.1, the observations specify a small fraction of data that are sufficient for CI-index forecasting using a random forest (RF) M at desired accuracy. Fig 3 illustrates how a running of a Type 4 system creates such a dataset with $\mathcal{D}$ contains four datasets $\{D_w, D_b, D_N, D_P\}$, for a water table, a basin table, a Nitrogen table, and a Phosphorus table. The augmentation $\oplus$ is implemented with a library of spatial joins [27], a common query that join two tables based on the tuple-level spatial similarity.

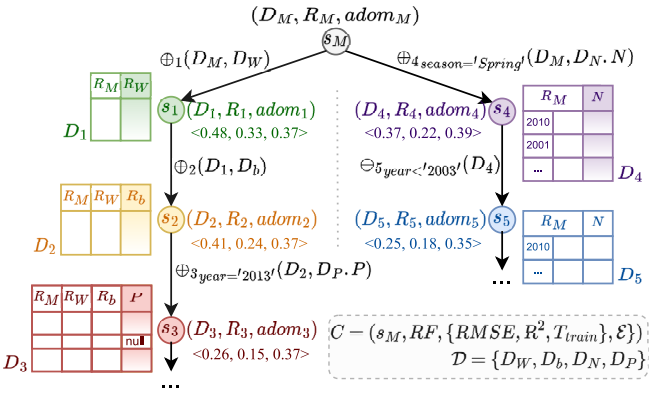

Figure 3: The running of a data discovery system, corresponding transition graphs, and result datasets.

With a configuration $C = (s_M, \text{RF}, \{RMSE, R^2, T_{train}\}, \mathcal{E})$, where $\mathcal{E}$ is an MO-GBM estimator $\mathcal{E}$, a running starts by joining D_w and D_b, the basin table and the upstream river data to get D_2. It then augments D_2 with one more attribute "Phosphorus" under a selection condition "year = "*2013*", and leads to result D_3, via a path with a label $\{\oplus_1, \ldots, \oplus_3\}$. In each step, a test is initialized (if not already in $\mathcal{T}$); and the estimator $\mathcal{E}$ is consulted to estimate the performance vector to enrich $\mathcal{T}$.

Similarly, another path is spawned by first converting D_M to D_4 with an augmentation, followed by a reduction that removes historical tuples older then "2003", that are estimated to be less relevant. Fig 3 depicts a fraction of the transition graph of the above running, containing the two paths.

3.2 Properties Analysis

We next discuss fundamental properties of a data discovery system. These properties help us justify the generality as well as practical implementations of data discovery algorithms (see Section 5).

Expressiveness. We study the expressiveness power of the data discovery system in terms of all the paths with operators as "labels" it can produce in all its possible transition graphs. This gives us a set of strings, forming its language $L(\mathcal{T})$. We show that a data discovery system and its runnings resembles, and can be specified in practice for data integration [16], or feature selection [20], We provide the following result.

Proposition 3.3. *A data discovery system $\mathcal{T}$ can be configured to express (1) data augmentation, and (2) feature selection.*

Proof sketch: We show the above cases by an analysis on the languages generated by the specifications of type 4 system $\mathcal{T}_4$. To see this, it suffices to show that (1) $L(\mathcal{T}_1) \subseteq L(\mathcal{T}_4)$, and $\mathcal{T}_1$ can express feature selection; and (2) $L(\mathcal{T}_2) \subseteq L(\mathcal{T}_4)$, and $\mathcal{T}_2$ can simulate data augmentation process. One can readily infer (1) and (2) by treating $\mathcal{T}_1$ and $\mathcal{T}_2$ as special cases of $\mathcal{T}_4$, with proper configurations. For (1), one removes the model M and constrain O as data reduction operators only without selection condition. For (2), one constraint O as data augmentation ("join") operators. □

Non-blocking Processing. We also verify a desirable property of data discovery computation. A data operator (query) op is *non-blocking* [15], if for any path ρ with a transition that applies op and a result D in the running of a transducer $\mathcal{T}$, it generates the same result D regardless of at which step it applies op in ρ (whenever applicable). We have the following claim.

Claim: *The operators $\oplus$ and $\ominus$ are often non-blocking in practice.*

We observe that in most applications, (a) $\oplus$ is commutative and associative, and (b) any consecutive application of an $\oplus$ operator followed by an operator in $\ominus$ are commutative. This ensures that the operators $\oplus$ and $\ominus$ are non-blocking. Moreover, this suggests that any paths in $\mathcal{T}$ are order-independent.

Following the above analysis, we have the result below.

Proposition 3.4. *Given a set of operators O with non-blocking operators $\oplus$ and $\ominus$, Type 3, Type 4 and Type 5 systems have the same expressiveness.* i.e., $L(\mathcal{T}_3) = L(\mathcal{T}_4) = L(\mathcal{T}_5)$.

In other words, any dataset that can be created by a running of a general form $\mathcal{T}_4$ can be simulated by a running of T_3 or T_5 under a proper configuration that leads to the same result. This allows us to choose a proper design of $\mathcal{T}$ from any of $\mathcal{T}_3$, $\mathcal{T}_4$ or $\mathcal{T}_5$ to perform the search with a completeness guarantee.

The above properties further indicate effective asynchronous implementation of data discovery in distributed environment for non-blocking operators. We defer such discussion in future work.

4 MULTI-OBJECTIVE DATA DISCOVERY

Given a configuration, we want to find a running of $\mathcal{T}$ that leads to a "global" optimal dataset, over which M is expected to achieve the best performance for all measures. Nevertheless, such optimal solution may not always exist. (1) Even normalized to be minimized, the measures in $\mathcal{P}$ may in nature conflict as trade-offs, such as training cost versus accuracy, or precision versus recall. (2) Given the "no free lunch" theorem [28], it is likely that no data-driven (ML) model perform better than others. We thus instead consider a scheme that pursues *Pareto optimality* for $\mathcal{D}_F$.

To this end, we introduce a state dominance relation below.

Dominance. Given $\mathcal{P}$ and a data discovery system $\mathcal{T}$, a state $s = (D_s, R_s, \text{adom}_s)$ is *dominated* by another state $s' = (D'_s, R'_s, \text{adom}'_s)$ (resp. s' dominates s), denoted as $s \prec s'$ (resp. $s' \succ s$), if there are two valuated tests $t = (M, D_s)$ and $t' = (M, D'_s)$ in T, such that

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

As each state s is associated with a dataset D_s, consistently, we say a dataset D'_s dominates another D_s, denoted as $D'_s \prec D_s$, if and only if the states $s' \prec s$.

Given $\mathcal{T}$ and a configuration C, let $\mathcal{D}_F$ be the set of all the possible output datasets that can be generated by a running of $\mathcal{T}$, a set of dataset $\mathcal{D}^*_F \subseteq \mathcal{D}_F$ is a *Pareto set w.r.t.* $\mathcal{T}$ and C, if

- any pair D_1 and D_2 in $\mathcal{D}^*_F$ cannot dominate each other; and
- for any other $D \in \mathcal{D}_F \setminus \mathcal{D}^*_F$, and any $D' \in \mathcal{D}^*_F$, $D \prec D'$.

We now formulate the multi-objective data discovery problem.

Problem Statement. Given a configuration $C = (s_M, M, T, \mathcal{E})$, and a data discovery system $\mathcal{T}$ that specifies non-blocking operators O and δ, the *multi-objective data discovery* problem, denoted as MODis, is to compute a Pareto set $\mathcal{D}_F$ in terms of $\mathcal{T}$ and C.

Example 4.1. Revisiting Example 3.2 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}
$t_1 : (D_1, \text{RF}, \mathcal{P}, MO-GBM)$	0.48	0.33	0.37
$t_2 : (D_2, \text{RF}, \mathcal{P}, MO-GBM)$	0.41	0.24	0.37
$t_3 : (D_3, \text{RF}, \mathcal{P}, MO-GBM)$	0.26	<u>0.15</u>	0.37
$t_4 : (D_4, \text{RF}, \mathcal{P}, MO-GBM)$	0.37	0.22	0.39
$t_5 : (D_5, \text{RF}, \mathcal{P}, MO-GBM)$	<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 measurement values 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 Pareto set $\mathcal{D}_F$ currently contains $\{D_3, D_5\}$, over the valuated cases.

The problem, in nature a multi-objective optimization problem, is in general intractable [7]. We next investigate feasible algorithms that implements the running of data discovery systems, with quality guarantees. We also investigate how different search strategies may impact the efficiency. Our main results are summarized in Table 2.

5 COMPUTING PARETO OPTIMAL DATASETS

To compute a Pareto set, a straightforward solution performs complete runnings of the system $\mathcal{T}$, to verify M's performance over all the N states (datasets) and invoke established multi-objective optimization methods such as Kung's algorithm [14]. This will incur $O(N \log N)^{|\mathcal{P}|-2}$ number of valuation (for $|\mathcal{P}| \geq 4$), or $O(N(\log N))$ if $|\mathcal{P}| < 4$. Such valuation cost is infeasible in practice, even when enlisting N states as a "once-for-all" cost is affordable. Moreover, a Pareto set may already contain an excessive number of datasets that are too expensive to be inspected.

We next present an algorithm that approximates Pareto set with a size-bounded solution, and reduces unnecessary valuations.

5.1 Approximating Pareto Optimality

To characterize a feasible algorithm for data discovery with size-bounded solutions and quality guarantee, we introduce a notion of ϵ-Pareto set, followed by an optimality measure.

ϵ-Pareto set. Consider a data discovery system $\mathcal{T}$ with a configuration $(s_M, \mathcal{O}, M)$. Let $\mathcal{D}_\mathcal{S}$ be a set of all the datasets generated by a running of $\mathcal{T}$. Given two datasets D and D' in $\mathcal{D}_\mathcal{S}$, and a constant $\epsilon > 0$, we say D *ϵ-dominates* D' *w.r.t.* M, denoted as $D \preceq_\epsilon D'$, if for the corresponding tests t=(M, D) and t'=(M, D'), $t.p \leq (1+\epsilon)t.p$ for each $p \in \mathcal{P}$.

A set of datasets $\mathcal{D}_\epsilon \subseteq \mathcal{D}_\mathcal{S}$ is an *ϵ-Pareto set* of $\mathcal{D}_\mathcal{S}$, if for every dataset $D' \in \mathcal{D}_\mathcal{S}$, there exists a dataset $D \in \mathcal{D}_\epsilon$ such that $D \preceq_\epsilon D'$.

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

- for arbitrarily given, and fixed small constant $\epsilon > 0$, it correctly outputs an ϵ-Pareto set *w.r.t.* N valuated states; and
- the time cost is a polynomial determined by $|\mathcal{D}|$, N, and $\frac{1}{\epsilon}$.

Example 5.1.

Below we present our main result.

Theorem 5.2. *Given a data discovery system $\mathcal{T}$, a configuration C=$(S_M, M, T, \mathcal{E})$, and a number N, there exists a (N, ϵ)-approximation that takes in total $O($**TBF**$)$ time.*

Remarks. The scheme characterizes a relative optimality *w.r.t.* ϵ and input size of states N. If N refers to the number of all possible states by an exhaustive running of $\mathcal{T}$, it *ensures* to output a ϵ-Pareto set of a Pareto set $\mathcal{D}^*_F$. Yet the algorithm is *not* in polynomial time *w.r.t.* dataset size $|\mathcal{D}|$ alone, as N can be exponential. On the other hand, it is still desirable as one can strike a balance between the 'closeness" of the output to a Pareto set, and the actual time cost, by explicitly tuning ϵ and a managable size N.

5.2 Constrained ϵ-Approximation

As a constructive proof of Theorem 5.2, we next present an algorithm, denoted as ApxMODis.

"Reduce-from-Universal". Algorithm ApxMODis implements a Type-5 system (Section 3.1) and starts with a node as a "universal" dataset D_U (with a universal schema). In a nutshell, it "transforms" state dominance to a "path dominance" counterpart, which aggregate valuated measures from nodes to the end of the paths at runtime. This strategy has the following desirable properties. (1) As we uniformly minimize all the measurements, this allows us to grow "shortest" paths with prioritized reductions and valuation to refine D_U towards any of user provided lower bounds, encouraging *early termination*; (2) starting from more features and more tuples allows smaller sacrifice of model accuracy and generality.

Auxiliary structure. Algorithm ApxMODis induces and maintains a transition graph $G_\mathcal{T}$ with N *unvaluated nodes*. During the runtime traversal of $G_\mathcal{T}$ with a path ρ, it also mantains a path label

traverse It enhances $G_\mathcal{T}$ with additional run-time structure as follows. (1) For each path

From transducer $\mathcal{T} = (s_M, \mathcal{S}, \mathcal{O}, \mathcal{S}_F, \delta)$, we formalize a transition graph $G_\mathcal{T} = (\mathcal{S}, \delta)$, where vertices $s \in \mathcal{S}$ represent states of the dataset D. Each vertex is labeled with a universal schema that includes the features' states. Start state s_M represents the state of the initial dataset D_M. By performing an operation $op \in \mathcal{O}$ on a state s, a new state s' is generated along with a transition $r = (s, op, s')$. The cost of this transition $\mathbf{c}(r)$ is determined by the performance variances between two tests $t = (M, D_s)$ and $t' = (M, D_{s'})$, denoted as $\mathcal{P}_i(r) = t.p_i - t'.p_i$, $\forall p_i \in \mathcal{P}$. A *path* $\rho = \{r_1, \ldots r_k\}$ is labeled with a tuple $(l(s_k), \mathcal{P}(\rho), pred(\rho))$, where $l(s_k)$ is the label of the last state s_k, $\mathbf{p}(\rho) = \sum_{r \in \rho} \mathcal{P}(r)$, and $pred(\rho)$ is the label of the *subpath* $\rho' = \{r_1, \ldots r_{k-1}\}$. We discretize the $\mathcal{P}(\rho)$ in each objective and compute the position for a path ρ in the Pareto set using Equation 1.

$$\text{pos}(\rho) = \left[\left\lfloor \log_{1+\epsilon} \frac{c_1(\rho)}{c_1^{\min}} \right\rfloor, \ldots, \left\lfloor \log_{1+\epsilon} \frac{c_{d_c}(\rho)}{c_{d_c}^{\min}} \right\rfloor, \left\lfloor \log_{1-\epsilon} \frac{b_1(\rho)}{b_1^{\max}} \right\rfloor, \ldots, \left\lfloor \log_{1-\epsilon} \frac{b_{d_b}(\rho)}{b_{d_b}^{\max}} \right\rfloor \right]^T \quad (1)$$

Consider $\mathcal{P}$ may encompass costs (*e.g.,* training time) and benefits (*e.g.,* accuracy), which we represent as c and b respectively, d_c and d_b as their total count. For each $c_i \in c$, we capture the base-$(1+\epsilon)$ logarithm of the ratio between the specific cost value $c_i(\rho)$ and

its minimum counterpart c_i^{min}, and then floor the result. Benefits are computed similarly to costs, with some modifications: the logarithmic base is $1-\epsilon$, and we normalize against $b_i^{\max}$. The position vector $pos(p)$ is constructed by combining the results of the costs and benefits and then scaled to $[0,1]$ as normalized $\mathcal{P}$ for path ρ, all $p \in \mathcal{P}$ aim to minimize. In ApxMODis, we calculate the position of a path based on its first $d-1$ objectives and set the d_{th} one as a deterministic metric to ensure the Pareto Set remains polynomial-size bounded, specifically within the dimensions of $[\mathbb{N}_0]^{d-1}$.

Algorithm. The algorithm ApxMODis is illustrated in Fig. 4, which induces a Pareto Set $\mathcal{D}_F$ from a data discovery system $\mathcal{T}$. In *procedure* ApxMODis, we first construct a transition graph $G_{\mathcal{T}}$ using states $\mathcal{S}$ and transitions δ, with a maximum depth of n. In line 5 to 8, for each state $s \in \mathcal{S}$, we initialize the initial layer of $\mathcal{D}_{F_s}$ as an empty set and compute all $p \in \mathcal{P}$ for the current state s over M. At line 9, we assign the label of path ρ_M, with length 0, to position $\text{pos}(\rho_M)$ in $\mathcal{D}_{F_{s_M}}$. The iteration in line 10 to 14 is refining the Pareto Set $\mathcal{D}_F$ at each depth. When we reach a certain depth, for each state $s \in \mathcal{S}$, we carry over the Pareto Set from the previous round, then check all possible transitions that lead to s by Extend-&-Merge function, and update $\mathcal{D}_{F_s}$ at the current depth accordingly. In *Extend-&-Merge function*, for each path ρ' that can potentially reach s, we generate $\rho = \rho' \cup \{s\}$ and evaluate it. If any of its measures in $\mathcal{P}$ exceed the thresholds in **ts**, we execute and proceed to the next path. Using Equation 1, we compute the ρ's position in the Pareto set, which will be a $(d-1)$-vector without considering the deterministic metric. In line 24 to 28, we check the dominance with the paths at $pos(\rho)$ over all states by comparing the deterministic metric. This ensures that each position in the global Pareto set will only hold one most optimal path and keep the size of $\mathcal{D}_F$ polynomially bounded. After iterating through all depths, the algorithm returns the refined Pareto set $\mathcal{D}_F$.

Approximability. Given a graph G with a set of states $\mathcal{S}$ and set of transitions δ, each transition $r = (s', \text{op}, s)$ will be assigned a set of performance measures $\mathcal{P}$, the multi-objective path search problem(MOPS) is to compute a Pareto set Π, which consists of a set of paths such that no path dominates others. Here a path ρ dominates another ρ', if its performance measures $\mathcal{P}(\rho)$, computed as $\sum_{r\in\rho}\mathcal{P}(r)$, dominates $\mathcal{P}(\rho')$, similarly as defined as in state dominance. By transforming an instance of MOPS to an instance of MODis, ApxMODis leverages an algorithm designed for MOPS to approximate solutions for MODis. Suppose an FPTAS schema exists for MOPS, we can also achieve a solution for MODis by ApxMODis that adheres to the same approximation quality.

Proof sketch: To prove the approximability of MODis, we construct an approximation preserving reduction from it to MOPS. We define a function f to convert an instance of MODis to MOPS in PTIME by constructing the transition graph $G_{\mathcal{T}}$, which we show in the Auxiliary structure. Then we map paths in Π from MOPS to dataset states using function g, forming the ϵ-Pareto Set Π' for MODis. The dominance principle ensures quality preservation. Based on this reduction, we show that ApxMODis provides an FPTAS schema for MOPS, which outputs an ϵ-Pareto set in polynomial time *w.r.t.* input size and $\frac{1}{\epsilon}$. This confirms the approximability of MODis. Detailed proof can be found in the Appendix. □

Algorithm 1 Algorithm ApxMODis

1: **input:** Initial dataset D_M, ϵ, maximum depth n, data discovery system $\mathcal{T} = (s_M, \mathcal{S}, \mathcal{O}, M, \mathcal{D}_F, \delta)$, thresholds on costs $\boldsymbol{ts}$
2: **output:** Pareto set $\mathcal{D}_F$
3: **procedure** ApxMODis($D_M, \mathcal{T}, n, \epsilon, \boldsymbol{t}$)
4: **construct** trans. graph $G_{\mathcal{T}} = (\mathcal{S}, \delta)$ **with** max depth n;
5: **for** each state $s = (D_s, R_s, \text{adom}_s) \in \mathcal{S}$ **do**
6: $\mathcal{D}_{F_s}^0[0] = \emptyset$;
7: **for** each metric $p \in \mathcal{P}$ **do**
8: $\mathcal{P}_s$.append($f(D_s, M).p$);
9: $\mathcal{D}_{F_{s_M}}^0[pos(\rho_M)] = \{((R_{s_M}, \text{adom}_{s_M}), \mathcal{P}_{s_M}, null, null)\}$;
10: **for** $i \leftarrow 1$ **to** n **do**
11: **for** all states $s \in \mathcal{S}$ **do**
12: $\mathcal{D}_{F_s}^i = \mathcal{D}_{F_s}^{i-1}$;
13: **for** all $r = (s', \text{op}, s) \in \delta$ **do**
14: $\mathcal{D}_{F_s}^i$ = Extend-&-Merge(s, $\mathcal{D}_{F_s}^i$, $\mathcal{D}_{F_{s'}}^{i-1}$, ϵ);
15: **return** $\mathcal{D}_{F_{s\in\mathcal{S}}}^n$;
16: **end procedure**
17: **function** Extend-&-Merge(s, R, Q, ϵ)
18: **for** each $\rho' \in Q$ **do**
19: $\rho = ((R_s, \text{adom}_s), \mathcal{P}_s, \rho', (s', \text{op}, s))$;
20: **for** $i \leftarrow 1$ **to** $|\mathcal{P}_s|$ **do**
21: **if** $\mathcal{P}_s[i] > \boldsymbol{ts}[i]$ **then** continue;
22: **compute** pos(ρ) **with** ϵ **by** Equation (1);
23: **set** p **as** deterministic metric, skip = False;
24: **for** $s' \in \mathcal{S}$ **do**
25: **if** $\mathcal{D}_{F_{s'}}^i[\text{pos}(\rho)].p > \rho.p$ **then** skip = True;
26: Continue;
27: **del** $\mathcal{D}_{F_{s'}}^i[\text{pos}(\rho)]$
28: **if** skip ≠ True **then** $R[\text{pos}(\rho)] = \rho$;
29: **return** R;
30: **end function**

Figure 4: ApxMODis**: Approximating Pareto Sets**

Cost Analysis. As we ensured that one position in the Pareto set will only hold one path, the *Space Complexity* equals the size of the array for the Pareto set. Assuming all performance metrics are costs, according to Equation 1, the Space Complexity is $O\left(\prod_{i=1}^{|\mathcal{P}|-1}\left(\left\lfloor \log_{1+\epsilon}\frac{p_i^{max}}{p_i^{min}}\right\rfloor + 1\right)\right)$. For *Time Complexity*, in main iteration (line 10 to 14), for each depth from 1 to n, we evaluate all possible incoming transitions into the state at that depth. In the Extend&Merge function, we may reach out to all paths in the Pareto set. So the Time Complexity is $O\left(n|D_U|^n \prod_{i=1}^{|\mathcal{P}|-1}\left(\left\lfloor \log_{1+\epsilon}\frac{p_i^{max}}{p_i^{min}}\right\rfloor + 1\right)\right)$. Let's denote P^{max} as the maximum value of $\frac{p_i^{max}}{p_i^{min}}$, given ϵ is small enough, so $\log(1+\epsilon) \approx \epsilon$, then Time Complexity is $O\left(n|D_U|^n\left(\frac{\log(P^{max})}{\epsilon}\right)^{|\mathcal{P}|-1}\right)$.

Algorithm 2 Algorithm BiMODis

1: **TBF**

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

5.3 "Bi-Directional": A Type-3 Implementation

We next introduce an optimized specification of MODis, which take advantage of the correlation of the objectives to achieve early detection of state dominance relation.

For the start state of the backward search, based on an empty setting, we add a minimal set of active domains to ensure that each class in the target is included in the result dataset for classification tasks.

Utility Correlation Graph. Defined as a signed (weighted) graph after Spearman correlation [32].

Parameterized Dominance.

Algorithm. The algorithm, denoted as BiMODis, uses **TBF** as auxiliary structures, is illustrated in Fig. 5.

Lemma 5.3. *Give the pruning lemma.*

Discussion on Estimators. The lower and upper bound for the performance of a surrogate model [21].

Lemma 5.4. *If an estimator the performance value, assume* $\forall estimator, P_i(accuracy) \leq (1+\epsilon)\theta \cdot P$*, then the actual approximation ratio would be* $(1+f(\epsilon))$

Practically show that it is small and cite some papers.

Exponential time, with a strong approximate guarantee. The size of the Pareto Set is polynomially bounded [29].

5.4 Diversified Data Discovery

TBFAdd motivation. We provide our second specification, which is able to generate datasets that are diversified, yet still provide a sub-optimality guarantee in terms of Pareto optimality. Our goal is to maximize the pairwise distance of the generated datasets $\mathcal{D}_F$:

$$\max_{D,D' \in \mathcal{D}_F} \sum_{i=1}^{k} \sum_{j=i+1}^{k} d(D_i, D_j)$$

The algorithm, denoted as DivMODis, is outlined below.

$$d(u,v) = a\frac{euclid(u.pos, v.pos)}{euclid_{max}} + (1-a)\frac{1-cos(u.lable, v.lable)}{2}$$

$$euclid_{max} = euclid(pos(c_{min}, b_{max}), pos(c_{max}, b_{min}))$$

6 MAINTENANCE OF DATASETS

We provide our main algorithms, specifications and guarantees in the following table.

7 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**: How fast they are in response to finding data for a model as "queries"? **RQ3**: What's the impact of factors such as data size, quality requirement, number of performance measures, and optimization strategies? We also illustrate the applications of our approaches, with case studies.

Datasets. We use the following datasets, summarized in Table 3.

(1) Kaggle [11]. T_3

(2) OpenData [1]. We used Open Data, one of the largest public data sets, and selected about 2K tables, which include schools recording, school evaluations and school types, houses recording, housing price information in New York and Chicago, geographical location and other real-world data. T_1, T_2, extend from METAM

(3) HF [10]. T_4, find a regression model from HF

Models and Performance Evaluation. We have trained the following models with the datasets consistently with the settings in [5, 6] (1) Two random forest models (RFhouse and RFschol), for classifying house price, and for school performance, respectively, using OpenData. (2) more models paired with datasets.

(3) We trained a Gradient Boosting Model (GBmovie) for predicting the movies' worldwide gross sale, using Kaggle data. more; give a table here.

We trained all these models with scikit-learn [24]. add more training settings. For a fair comparison, we use the original training scripts from the baseline methods and validated that the retrained models have comparable original performance as reported the baseline methods.

Algorithms. We compare the performance with the following approaches, all in Python.

- **MODis**: Our multi-objective solutions, including ApxMODis, BiMODis, and DivMODis. In addition, we implemented NOMODis, a counterpart of BiMODis without the optimization strategy.
- METAM [6]: A goal-oriented data discovery framework that queries a downstream task with candidate dataset based on dataset utility. Although it attempts to find optimal augmentation, its performance in discoverying more relevant data is limited and cannot satisfy users who want more data augmentation.
- Starmie [5]: An end-to-end framework for dataset discovery from data lakes with table union search as the main use case. However, they main idea is table-level augmentation search, and with longer contrastive model training time.
- SkSFM [24]: add description. Automated feature selection with scikit-learn's SelectFromModel method, a simple wrapper, selects features using a built-in estimator.
- H2O [8] add description.

Evaluation. We evaluate data discovery algorithms in terms of task scenarios. We created different tasks, summarized in Table **??**. For each task, we initialized our algorithms with corresponding configuration that specifies original dataset D_M (associated with start state s_M), the trained model M, and model performance metrics $\mathcal{P}$. All the baseline methods only consider a single performance measure, which are underlined in the table.

Setting	Complexity	Algorithm	Approx. ratio	Time Cost
Multi-Objective (MO)-Data Discovery	NP-hard	ApxMODis	$1+\epsilon$	
		BiMODis	$1+\epsilon$	
Diversified MO-Discovery	NP-hard	DivMODis		
Incremental MO-Discovery	NP-hard	IncMODis		

Table 2: Problems, Hardness, and Guarantees TBF

Dataset	#tables	#Columns	#Rows	Size
Kaggle	1943	33573	7317773	704.6MB
OpenData	2457	71416	33296998	8.58GB
HF				

Table 3: Characteristics of Datasets

Task	Dataset	Model	Perf.	Note
House Price (C)	OpenData	RFhouse	$\mathcal{P}_1$	
School Perform.(C)	OpenData	RFschol	$\mathcal{P}_2$	
Movie Gross (C)	Kaggle	GBmovie	$\mathcal{P}_3$	
Avocado Price (R)	HF	LRavocado	$\mathcal{P}_4$	

Table 4: Task scenarios (Configurations)

Evaluation metrics. We adopt the following metrics to quantify the effectiveness of data discovery approaches. Denote as D_M an initial dataset, and $\mathcal{D}_o$ a set of output datasets from a data discovery algorithm. (1) Both METAM and Starmie generate a single dataset D_o for a designated task-oriented metric p. By default, p is **TBF** for regression task, and **TBF** for classification. (2) For MODis algorithms, we set $\mathcal{P}$ to contain the measure p accordingly as one of **TBF** or **TBF**, with 5 additional normalized measures: **TBF**, **TBF**, **TBF**, **TBF**, and **TBF**. Given the output of MODis algorithms $\mathcal{D}_o$, we choose 6 datasets D_{p_i} such that each has the highest estimated model performance for metric $p_i \in \mathcal{P}$ ($i \in [1,6]$). (3) We define the *relative model improvement* $\mathsf{rImp}(p)$ for a given model performance measure p achieved by a data discovery method as $\frac{M(D_M).p}{M(D_p).p}$. As all metrics are normalized to be minimized, the larger $\mathsf{rImp}(p)$ 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 total time cost of the data discovery process upon receiving a given model or task as a "query".

Task scenarios. We created the following task scenarios with specified task, model, and performance measures (Perf.), summarized in Table 4. Give the table. Here $\mathcal{P}_1$ - $\mathcal{P}_3$ are defined as follows. No need to use the same set of performance metrics. Anything larger than one is good. Accordingly for radar graph – you can have one with Axis of three, four, or five.

Exp-1: Effectiveness.

Effectiveness: single performance measure. Add a table

Effectiveness: multiple performance measures. Show in Fig. 6.

Impact factors. Show in Fig. 7.

Impacts of ϵ and maximum length, v.s. accuracy for a classification task and RMSE for a regression task.

As a method for Table Union Search(TUS), Starmie is effective at finding related tables in a data lake, **TBF**. However, its algorithm does not consider measures from a downstream data science task.

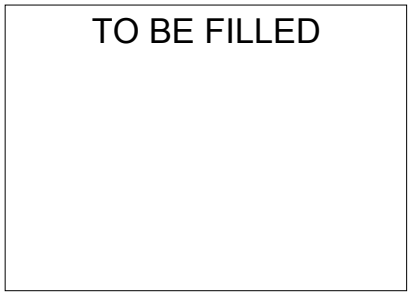

Figure 6: Effectiveness Radar Graphs

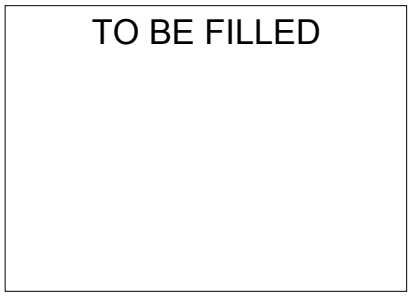

Figure 7: Impact of factors

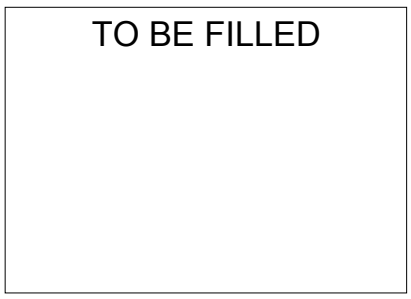

Figure 8: Efficiency

So it is in expect that **TBF**. This highlights the importance of goal-driven data discovery.

Exp-2: Efficiency.

Varying # of attributes.

Varying $|\mathcal{D}|$.

Varying ϵ.

Varying path length.

Show in Fig. 8.

Exp-3: Case study.

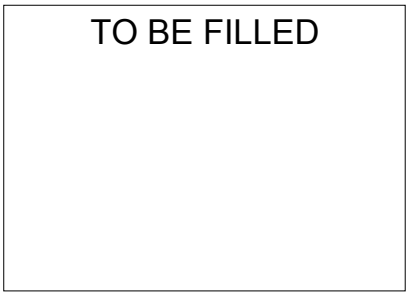

Figure 9: Case Study

"Search data with model". Show in Fig. 9. Find a case from HF, how this will help suggest a proper dataset for a model/script.

Data generation for Model Evaluation. Our algorithm generates a set of recommended datasets based on a model and user-defined metrics with expected ranges in just one round.

8 CONCLUSION

REFERENCES

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

[2] Dan Brickley, Matthew Burgess, and Natasha Noy. 2019. Google Dataset Search: Building a search engine for datasets in an open Web ecosystem. In *The World Wide Web Conference*. 1365–1375.

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

[4] European Organization For Nuclear Research and OpenAIRE. 2013. Zenodo. https://doi.org/10.25495/7GXK-RD71

[5] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée Miller. 2022. Semantics-aware Dataset Discovery from Data Lakes with Contextualized Column-based Representation Learning. *arXiv preprint arXiv:2210.01922* (2022).

[6] Sainyam Galhotra, Yue Gong, and Raul Castro Fernandez. 2023. METAM: Goal-Oriented Data Discovery. *arXiv preprint arXiv:2304.09068* (2023).

[7] Christian Glaßer, Christian Reitwießner, Heinz Schmitz, and Maximilian Witek. 2010. Approximability and hardness in multi-objective optimization. In *Programs, Proofs, Processes: 6th Conference on Computability in Europe, CiE 2010, Ponta Delgada, Azores, Portugal, June 30–July 4, 2010. Proceedings 6*. 180–189.

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

[9] José Hernández-Orallo, Wout Schellaert, and Fernando Martínez-Plumed. 2022. Training on the test set: Mapping the system-problem space in AI. In *AAAI*, Vol. 36. 12256–12261.

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

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

[12] Aamod Khatiwada, Roee Shraga, Wolfgang Gatterbauer, and Renée J Miller. 2022. Integrating Data Lake Tables. *Proceedings of the VLDB Endowment* 16, 4 (2022), 932–945.

[13] Ron Kohavi and George H John. 1997. Wrappers for feature subset selection. *Artificial intelligence* 97, 1-2 (1997), 273–324.

[14] 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.

[15] Yan-Nei Law, Haixun Wang, and Carlo Zaniolo. 2004. Query languages and data models for database sequences and data streams. In *VLDB*. 492–503.

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

[17] Guoliang Li. 2017. Human-in-the-loop data integration. *Proceedings of the VLDB Endowment* 10, 12 (2017), 2006–2017.

[18] 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.

[19] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. 2018. Accelerating machine learning inference with probabilistic predicates. In *Proceedings of the 2018 International Conference on Management of Data*. 1493–1508.

[20] Jianyu Miao and Lingfeng Niu. 2016. A survey on feature selection. *Procedia computer science* 91 (2016), 919–926.

[21] Mohammadreza Mousavi Kalan, Zalan Fabian, Salman Avestimehr, and Mahdi Soltanolkotabi. 2020. Minimax lower bounds for transfer learning with linear and one-hidden layer neural networks. *Advances in Neural Information Processing Systems* 33 (2020), 1959–1969.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

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

[29] 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.

[30] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. 2021. Logme: Practical assessment of pre-trained models for transfer learning. In *International Conference on Machine Learning*. 12133–12143.

[31] Huaqing Zhang, Jian Wang, Zhanquan Sun, Jacek M Zurada, and Nikhil R Pal. 2019. Feature selection for neural networks using group lasso regularization. *IEEE Transactions on Knowledge and Data Engineering* 32, 4 (2019), 659–673.

[32] 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.

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

APPENDIX: PROOFS AND ALGORITHMS

Proof of Theorem 5.2 *Given a data discovery system $\mathcal{T}$ and a configuration C=(S_M, M, T), (1) there exists a fully polynomial time approximation scheme (FPTAS) for multi-objective data discovery problem; (2) the scheme takes in total $O($**TBF**$)$ time.*

Proof:

(1) To show there exists an FPTAS for the multi-objective data discovery problem (MODis), we first constructed an approximation preserving reduction from it to a multi-objective shortest path search problem (MOPS).

Given an instance I of MODis, which comprises a data discovery system $\mathcal{T}$ and a configuration C=(S_M, M, T). Here, s_M is the initial state of a dataset D, and we have a transformation function g that constructs a graph $G = (V, E)$ in *polynomial time* (PTIME). In G, each vertex represents a unique state s of D, with s_M being the source node, and each edge represents a transition $r = (s, op, s')$ between states s and s'. For each edge $e = (s, s') \in E$, we generate two tests over M, labeled as $t = (M, D_s)$ and $t' = (M, D_{s'})$. The weight $\mathbf{c}(e)$ is calculated based on the variance in performance measures $\mathcal{P}$ over M from s to s', we represent this as $\mathbf{w}(e) = f(t).\mathcal{P} - f(t').\mathcal{P}$. Such that, the objectives in MOPS are the sum of weights for all edges in a given path ρ in G, denoted as $\mathbf{c}(\rho) = \sum_{e \in \rho} \mathbf{w}(e)$. An oracle may be utilized to obtain P under the assumption of 100% estimation accuracy within PTIME. Thus, I is transformed to $g(I)$, which is an instance of MOPS.

For any solution to MOPS, which produces an ϵ-Pareto Set Π of paths in G, we introduce a PTIME transformation function h that

maps the end vertex of the paths in Π to corresponding dataset states, forming the ϵ-Pareto Set Π' for MODis. In MOPS, path ρ dominates ρ' if and only if all objectives of ρ are superior to those of ρ'. After translating to MODis using h, it can be inferred that state s resulting from ρ is more desirable than state s' resulting from ρ' if all costs of s are smaller. This consistency guarantees that the quality of both Π and Π' is preserved.

Therefore, there is an L-Reduction from MODis to MOPS.

(2) Next, we prove Algorithm 1 is an FPTAS for MOPS.

Correctness. We prove the correctness by induction. (i) In the first iteration($i = 1$), $p \in \Pi_s^0$, if $\forall v \in V, \exists q \in \Pi_v^1$, such that $pos(q) = pos(p)$, and $c_d(q) \leq c_d(p)$. According to Equation 1, we have $\lfloor \log_{r_k} \frac{c_k(q)}{c_k^{\min}} \rfloor = \lfloor \log_{r_k} \frac{c_k(p)}{c_k^{\min}} \rfloor, \forall k \in [1, d)$, therefore, $\log_{r_k} \frac{c_k(q)}{c_k^{\min}} - 1 \leq \log_{r_k} \frac{c_k(p)}{c_k^{\min}}$. Along with $r_d = 1$, we have $c_k(q) \leq r_k c_k(p), \forall k \in [1, d]$. (ii) In the i_{th} iteration, we consider a path $p = (e_1, e_2, \ldots, e_l = (u, v)) \in \Pi_v^i$, where $l \leq i$, and a subpath $p' = p/e_l$. Assuming that $\exists q' \in \Pi_u^{i-1}$, such that $c_k(q') \leq r_k^{i-1} c_k(p'), \forall k \in [1, d]$. Then for $q'' = q' + e_l$, we have $c_k(q'') \leq r_k^{i-1} c_k(p), \forall k \in [1, d]$. Moreover, after the i_{th} iteration, $\exists q \in \Pi_v^i$, such that $pos(q) = pos(q'')$, and $c_d(q) \leq c_d(q'')$. Similar to (i), we have $c_k(q) \leq r_k c_k(q''), \forall k \in [1, d]$. Then, we can get $c_k(q) \leq r_k^i c_k(p), \forall k \in [1, d]$. Combine (i) and (ii), as $r_k = 1 + \epsilon_k$, in the special case, $\epsilon_k = \epsilon, \forall k \in [1, d)$, we ensure to get a ϵ-Pareto Set after the terminate iteration.

Cost Analysis. We next prove the exhaust searching time is polynomial in terms of n, m, and $1/\epsilon$, where n is the maximum path length and m is the number of edges in the given graph G. In Algorithm 1, we will run up to n iterations and check up to m edges per iteration for all possible filled positions in the last round's Π. So, the total time cost is

$$O\left(nm \prod_{j=1}^{d_c+d_b-1} \left(\lfloor \log_{r_j} nC_j \rfloor + 1\right)\right), C_i = \frac{c_i^{\max}}{c_i^{\min}} \vee C_i = \frac{b_{i-d_c}^{\min}}{b_{i-d_c}^{\max}}$$

Let's set $C = \max_{k \in [1,d)} C_j, d = d_c + d_b$ and $\epsilon_k = \epsilon, \forall k \in [1, d)$, then the time complexity is $O\left(nm \left(\frac{n \log nC}{\epsilon}\right)^{d-1}\right)$.

Since we solved MOPS by an FPTAS in total $O(\mathbf{TBF})$ time and L-reduction ensures approximation quality of the solutions is preserved, we can get there exists an FPTAS for MODis, which takes in total $O(\mathbf{TBF})$ time.

□