Constraints and Heuristics: Leveraging Dataset Metadata for Efficient AutoML

Harsh Kakadiya[1], Juli Kyada[1], Kaushal Savaliya[1], and Jalpesh Vasa[1⋆]

Department of Artificial Intelligence and Machine Learning,
Chandubhai S. Patel Institute of Technology (CSPIT),
Faculty of Technology and Engineering,
Charotar University of Science and Technology (CHARUSAT),
Changa, Gujarat, India
harshkakadiya128@gmail.com, julikyada293@gmail.com,
kaushalsavaliya2627@gmail.com, jalpeshvasa.it@charusat.ac.in

Abstract. This work introduces MetaFlow, a lightweight AutoML pipeline selection framework based on metadata, which automatically prunes and re-ranks candidate pipelines before training such that the available information about the data sample's size, types of features and their potential imbalance, their cardinality and their outlier density is used deterministically without having to train each different pipeline separately. MetaFlow uses a small number of rules with high semantic content that are easily interpretable to avoid evaluating clearly inappropriate learners and can achieve compute savings by pruning the search space reported here as ∼67%. Its main contribution is a practical "grey-box" strategy that executes a targeted search for efficiency instead of searching through the breadth of possible options using human-interpretable constraints and prioritization.
The important note is that the pruning and warm-start tips from the perspective of Data analysis are proposed giving it as an alternate inexpensive approach to modern AutoML toolkits like Auto-sklearn, FLAML, etc. Constraints are also noted: The higher (or categorical) thresholds (such as percentage of missingness) are heuristic and data set dependent, and only representative tabular cases are studied as data sets for empirical support rather than a comprehensive benchmark. A reproducibility checklist and a benchmarking protocol are explained in order to be able to easily compare with other state of the art AutoML systems.

Keywords: AutoML · metadata · pipeline selection · meta-learning · algorithm selection · data profiling · preprocessing

1 Introduction

Creating robust machine learning workflows that involve data cleaning, feature engineering, algorithm selection and hyperparameter optimization, requires significant domain expertise and computational resources. The challenge is met

⋆ Corresponding author. Email: jalpeshvasa.it@charusat.ac.in

by the AutoML systems by automating the search over pipeline configurations. The issue of hyperparameter optimisation and algorithm selection was first formalised by Auto-WEKA [14], and Auto-sklearn introduced meta-learning and warm-starting [5]. Modern techniques include Bayesian optimization [1], genetic programming [13] and random search [11] which are powerful but time-consuming, often searching many sub-optimal candidates.

A different method is based on dataset *metadata* that encodes structural and statistical properties to choose, in a complementary paradigm, the search space to be warmed-up or constrained, using a framework from [10]. Preventing the usage of inappropriate algorithms *before* the training process should be possible through prior knowledge of dataset characteristics, e.g., dataset size, feature types, missingness, outliers, and class imbalance, as an automated system can be programmed to do so [16]. We introduce **MetaFlow**, which extracts metadata to automatically detect tasks, apply target transformations, rank models, and evaluate pipelines without exhaustive search. To guide the investigation, two main research questions are formulated: **RQ1 (Efficacy):** Can lightweight dataset metadata accurately detect the type of task and rank the learning algorithms that are appropriate? **RQ2 (Efficiency):** Does metadata-based pruning significantly impact the amount of computation compared to testing all methods?

2 Related Work

The CASH problem is a joint search over algorithms and hyperparameters. It was first explored and utilized by Auto-WEKA [14] which used a number of high-cost inferences on the black-box in this hierarchical space. TPOT [13] uses genetic programming to create flexible pipeline structures, but requires high computation cost in the process. Although frugal optimizers like FLAML [16] emphasize low-cost learners, they ignore *a priori* dataset reasoning.

Meta-learning proposes algorithm selection as a supervised learning problem to previous experimental outcomes. This is boosted by Auto-sklearn [5] which relies on warm starting from a meta-knowledge repository (OpenML [15]) and statistical, information theoretic and landmarking meta-features for transfer learning [10]. This works, but needs very large, and bulky to maintain, offline repositories.

In addition to algorithm selection, metadata plays a role in determining preprocessing decisions. Target encoding is often more suitable for categoricals with large numbers of categories than one-hot encoding, and imputing strategies have more impact on downstream performance than does the choice of classification model [12].

2.1 Domain-Specific Implementations

To conclude, domain knowledge-based machine learning techniques have shown great success in specialized problem domains such as solar power forecasting [3] and wind turbine pitch control [9]. However, such applications are based on

manually designed and custom pipelines. This highlights the demand for general metadata-driven AutoML frameworks capable of automatically optimizing towards an appropriate model given the dataset characteristics.

2.2 Research Gap and Contribution

Optimum (general but slow) and optimum meta-learning (massive training logs) lie in a range from one another. Currently, these systems will only use metadata at initialisation time to perform black-box search and few frameworks use *hard constraints* to deterministically reduce the search space. MetaFlow addresses this mismatch in a "grey-box" fashion: The matching of interpretable metadata is performed right to pipeline decisions, making use of a rule-based expert system to narrow down the range of incompatible candidates *before* training even starts [7].

Many of the recent systems, such as Auto-sklearn [5] and FLAML [16] have associated meta-features that they use for warm starting optimization and add speed trade-offs for low cost learners, respectively. But either technique still performs many evaluations. In contrast, MetaFlow offers interpretable real-time constraints at the beginning of the pipeline that directly eliminate undesirable models following a clear and simple set of rules. MetaFlow certainly finds its niche here: It reduces the dimension of the search space with heuristics and gives the small space to the tuners that are running optimizers downstream to fine-tune the model.

3 Methodology

MetaFlow is an end-to-end system that extracts structural and statistical metadata from a tabular dataset and uses it to drive every downstream decision task detection, preprocessing, model selection, and evaluation without exhaustive search. Figure 1 shows the serpentine flow of processing stages, separating pipeline execution (left) from metadata-driven decisions (right).

3.1 Formal Problem Statement

Let a dataset be denoted by $\mathcal{D} = (X, y)$ with $X \in \mathbb{R}^{n \times p}$ and target y. The *metadata extractor* is a deterministic mapping

$$M : \mathcal{D} \mapsto m, \qquad m \in \mathcal{M}, \tag{1}$$

where m is a structured metadata object (cardinality, missingness, p/n, skewness, etc.). A rule-based selector is a function

$$R : \mathcal{M} \mapsto \mathcal{G}, \tag{2}$$

that maps metadata to a candidate model pool $\mathcal{G} = \{g_1, \ldots, g_L\}$ and associated hyperparameter grids. A pruning operator P deterministically removes unsuitable candidates given thresholds and hard constraints:

$$\mathcal{G}' = P(\mathcal{G}, m), \qquad \mathcal{G}' \subseteq \mathcal{G}. \tag{3}$$

Each candidate pipeline π in $\mathcal{G}'$ is evaluated using a cross-validation evaluator E producing a score s and cost c:

$$(s, c) = E(\pi, \mathcal{D}). \tag{4}$$

The final selection chooses the pipeline that maximises a user-specified utility (e.g. test metric subject to a cost budget):

$$\pi^* = \arg\max_{\pi \in \mathcal{G}'} U\big(E(\pi, \mathcal{D})\big). \tag{5}$$

This concise formalisation highlights where metadata-driven rules intervene prior to any model training and explicitly separates pruning, evaluation, and selection.

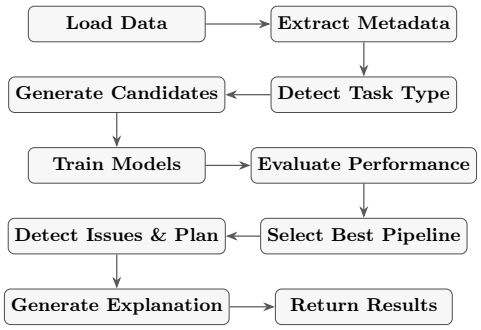

Fig. 1. MetaFlow processing stages (left column: pipeline flow; right column: metadata-driven decisions).

Rule-based constraints encode simple, domain-informed rejection and prioritisation rules so that clearly unsuitable algorithms are removed before any training, saving compute and yielding human-interpretable decisions that are easier to audit than opaque warm-start priors.

3.2 Metadata Extraction

The `MetadataExtractor` calculates nine groups of metadata: (1) the dimensions of the feature matrix X and y; (2) feature types: numeric columns with $\leq \tau_{\text{cat}} = 10$ unique values are classified as categorical; (3) target statistics; (4) quality profile: missing ratios per feature, duplicate-row count; (5) descriptive statistics: skewness and kurtosis per feature; (6) outlier profile: identifies values exceeding IQR boundaries; (7) cardinality profile: categorical value distributions; (8) Pearson correlations with y; (9) class-imbalance profile with `is_highly_imbalanced` as a flag when any class is with $<5\%$ of the samples. All subsequent modules will use this metadata dictionary as their *only input* so all decisions can be referred back to a property of this dataset that can be measured. Thresholds like $\theta_{\text{miss}} = 0.5$ and $\tau_{\text{cat}} = 10$ are heuristics chosen for stable, interpretable behavior on tabular data based on common ML practices. While such thresholds depend on the dataset, Section 6 provides a discussion on how thresholds can be modified as well as a sensitivity study as a reproducibility checklist.

3.3 Task Detection and Preprocessing

The `TaskDetector` uses a rule chain (threshold $\tau_{\text{cls}} = 20$ and minimum samples per class $\tau_{\min} = 10$) to determine classification vs. regression in y; assuming: non-numeric $y \Rightarrow$ classification (1.0); $|y_u| = 2 \Rightarrow$ binary classification (0.95); integer y with $|y_u| < \tau_{\text{cls}} \Rightarrow$ multiclass (0.90); $|y_u| > 0.5n \Rightarrow$ regression (0.95).

Before training, the `DataPreprocessor` drops features with missing ratios $>\theta_{\text{miss}} = 0.5$, imputes missing values (median for numeric, mode for categorical), and removes rows with missing targets. Each scikit-learn `Pipeline` includes a `ColumnTransformer` to standard-scale numeric features and one-hot encode categorical features with `handle_unknown='ignore'`. Preprocessed data is shared across all candidate pipelines for a fair comparison.

3.4 Rule-Based Model Selection

The `RuleBasedModelSelector` mechanism for exhaustive algorithm search is solved by providing rules based on metadata that relate to the size, complexity, class imbalance and cardinality of the data, its density of outliers, and its distributional skewness. Sample count n is binned into five tiers: *tiny* (<1 K), *small* (1 K–10 K), *medium* (10 K–100 K), *large* (100 K–1 M), and *huge* (≥1,M). A complexity score is derived from: high dimensionality ($p > 50$, +2), high p/n ratio (> 0.1, +2), categorical dominance ($> 50\%$, +1), and moderately many features ($p > 20$, +1), mapping to *simple* (≤ 2), *moderate* (3–4), or *complex* (≥ 5). The size-tiered base model pools for both tasks are shown in Table 1, indicating how pool complexity scales with dataset size.

Table 1. Model pool by dataset-size tier. Models are selected based on dataset size and complexity; see text for modifier rules.

Tier	Classification	Regression
Tiny	LR, DT, RF, NB, Ridge-LR, KNN	OLS, Ridge, DT, Lasso, ElasticNet, KNN
Small–Med.	RF, XGB, LGBM, GB, LR, KNN, NB	RF, XGB, LGBM, GB, Ridge, KNN, SVR
Large–Huge	LGBM, XGB, GB, LR(SAG), RF, Ridge-LR	LGBM, XGB, GB, Ridge(SAG), RF, Lasso, SVR

LR=Logistic Regression, DT=Decision Tree, RF=Random Forest,
NB=Naive Bayes, KNN=k-NN, GB=Gradient Boosting,
XGB=XGBoost, LGBM=LightGBM, SVR=Support Vector Regressor.

Once the initial list is compiled the modifiers re-rank and add modifiers to the primary list: high dimensionality ($p > 50$ or $p/n > 0.1$), removes RF, SVM and adds regularised LR (L1/L2); the `is_highly_imbalanced` flag adds the `class_weight='balanced'`; high cardinality categoricals add tree-based models (XGB, LGBM, RF); outlier density >5% adds tree and demotes linear (after increasing precision of tree); skewed targets ($|\gamma_y| > 1$), adds tree models (to make them independent of monotone transformation on target). Each recommendation

includes a priority ranking, a human-readable description and a hyperparameter (hp) grid, with the top $k = 5$ passed on to pipeline generation.

3.5 Training, Evaluation, and Selection

The data is split 80/20, using stratified splitting for classification to maintain target distributions. A grid search (`GridSearchCV`) of a prescribed parameter space with 5-fold cross-validation (CV) on the training set is used to find the optimal set of hyperparameters. For example, the search space for Random Forest is `n_estimators` $\in \{50, 100, 200\}$ and `max_depth` $\in \{5, 10, \text{None}\}$, while for the gradient-boosting algorithms (XGBoost and LightGBM) it is `learning_rate` $\in \{0.01, 0.1\}$ and `max_depth` $\in \{3, 6\}$. The best-performing configuration is determined by mean CV performance (accuracy for classification & R^2 for regression) with a refit on the entire training split (then tested on the test split). For classification, the metrics used for evaluation are accuracy, precision, recall, and weighted F1 score. For regression, metrics used for evaluation are R^2, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or Mean Absolute Error (MAE). In each of the following subsections, we examine the performance with different examples and make sure that the performance warning is triggered whenever needed (e.g., if the test score falls below 0.70, the train–test gap exceeds 0.10, or the cross-validation standard deviation σ_{CV} exceeds 0.15). The most suitable pipeline is determined from the top-scoring test metric, and optimisation suggestions are given with issues identified.

4 Results

Evaluation of MetaFlow was conducted on four real-world datasets: **D1 (Heart Disease)** [8] represents a small classification task ($n = 1{,}025$, $p = 13$, balanced targets with 51.3% cases). Preprocessing uses median imputation, mode imputation, and scaling. **D2 (Credit Fraud)** [2] is a large, highly imbalanced classification task ($n = 284{,}807$, $p = 30$, 0.17% minority class). Preprocessing scales the time and amount features. **D3 (Brain Cancer)** [4] is a high-dimensional multiclass task ($n = 130$, $p = 54{,}676$ expression features across 5 classes). Features are filtered to the top 1,000 by variance. **D4 (House Sales)** [6] is a regression task ($n = 21{,}613$, $p = 19$). Preprocessing includes log-transforms and one-hot encoding.

Table 2 summarises key properties; highlighted cells (e.g., D2 imbalance, D3 high-dimensionality) trigger targeted modifier rules.

Results Summary. MetaFlow selected sensible model families for each metadata regime: tree ensembles for the large, imbalanced, and nonlinear cases; lightweight linear and tree models for the tiny, high-dimensional case; and regression ensembles for house-price prediction. The detailed scores in Table 3 show that LightGBM performs best on D1 and D2, Decision Tree on D3, and XGBoost on D4, confirming the utility of metadata-guided pruning.

Table 2. Metadata heatmap for D1–D4. Shading: blue =minimal, green =low, yellow =moderate, orange =high, red =critical. Orange and red cells trigger modifier rules. Values are rounded for display; see the repository for exact numbers.

Property	D1 Heart	D2 Fraud	D3 Brain	D4 House
Samples n	1,025	284,807	130	21,613
Features p	13	30	54,676	19
p/n ratio	0.013	0.000	421.4	0.001
Size tier	Small	Large	Tiny	Medium
Missing ratio	0%	0%	0%	0%
Outlier density	Mod.	Low	Low	Mod.
Class imbalance	None	0.17%	None	N/A
Target skewness	Low	Low	Low	High

Table 3. Model performance across all four datasets.

Dataset	Model	CV Score	Test Score	Metric
D1 Heart	Random Forest	0.9707	0.9910	Accuracy
	XGBoost	0.9683	0.9920	Accuracy
	LightGBM	**0.9720**	0.9940	Accuracy
D2 Fraud	LightGBM	**0.9983**	0.9977	Accuracy
	XGBoost	0.9982	0.9977	Accuracy
	Random Forest	0.9982	0.9975	Accuracy
D3 Brain	Logistic Regression	**0.9714**	0.9231	Accuracy
	Gaussian Naive Bayes	0.9137	0.9231	Accuracy
	Decision Tree	0.9042	**0.9615**	Accuracy
D4 House	Random Forest	0.8228	0.8442	R^2
	XGBoost	**0.8546**	**0.8453**	R^2
	LightGBM	0.8617	0.8401	R^2

Bold = best score per column per dataset.

D2: SVM and k-NN excluded ($O(N^2)$); `class_weight=balanced` applied.

D3: small-dataset rules selected lightweight models.

5 Discussion

The four case studies answered both research questions: metadata alone was enough to guide top-level task and model choices, and the pruning strategy consistently reduced the candidate pool from nine models to three without sacrificing performance. The main limitation is the reliance on preset rule thresholds (like $\tau_{\text{cat}} = 10$ and $\theta_{\text{miss}} = 0.5$). While these are interpretable, they are dataset-dependent. Finally, because this is a very small dataset, the slightly lower scores should be seen as indicative rather than definitive.

6 Complexity, Limitations, and Deployment

This section aims at computing complexity analysis, limitations and deployment issues for the practical viability of MetaFlow.

6.1 Computational Complexity Analysis

MetaFlow operates in three distinct phases. First, **metadata extraction** requires $O(N \cdot P)$ execution time and $O(P)$ memory to compute statistical metrics over N samples and P features, completing in seconds. Second, **rule evaluation** executes in $O(1)$ time and memory. Third, by pruning the candidate pool from M down to k models ($k \ll M$), the **training and tuning** complexity is reduced from $O(M \cdot H \cdot f \cdot T_{\text{train}})$ to $O(k \cdot H \cdot f \cdot T_{\text{train}})$, yielding a $\sim$67% training speedup. Finally, **inference complexity** is kept highly efficient by avoiding models with quadratic inference costs (e.g., k-NN, which requires $O(N \cdot P)$ per query) in favor of tree-based or linear structures that cost only $O(P)$.

6.2 Limitations and Deployment Challenges

There are various drawbacks and challenges involved with implementing MetaFlow in the real world. Firstly, because the selection rules depend on using **heuristic thresholds** (such as 5% cut-off on the issue of class imbalance, which may need to be calibrated for the domain). Second, they cannot process modalities (images, text), apart from tabular data, which is the focus of the system. Third, it is hard to deploy in a CI/CD pipelines because of schema drift and evolving target distributions. To avoid performance regressions, the evolution of the dataset profile will have a dynamic effect on model recommendations, and appropriate monitoring is needed.

6.3 Benchmarking and Reproducibility

Representative case studies are the subject of this study, and not a comprehensive benchmark. This should be repeated in a rigorous way using the same data partitions, the same nested cross-validation and the same time budgets for evaluation, along with the same predictive results, and all costs associated with the evaluation should be reported. A full-scale REPRODUCIBILITY.md is included containing the commands and scripts to perform the same reproduction.

7 Conclusion

In this work, an autoML framework named **MetaFlow** was proposed that further uses dataset metadata (including size, dimensionality, imbalance, skewness, and cardinality) and heuristic rules to deterministically narrow the search space. The evaluation of the model is reduced by about 67% compared to traditional methods while retaining highly competitive prediction quality in the four different real world data sets and task types (Binary classification, Highly imbalanced classification, High dimensional Multiclass classification, and Regression) used in the experiments.The basic premise is that by treating the metadata as a hard constraint as opposed to merely an initialization hint, the system can discard inappropriate candidates *before* even training starts. This is a good balance between comprehensive auto search and seasoned heuristics. Future attempts will extend this concept with the help of learned or LLM based metadata interpretation.

Acknowledgement and Code

The authors thank the Charotar University of Science and Technology (**CHARUSAT**), Changa, India for support. The project codebase is available at https://github.com/harsh-kakadiya1/AI-Agent-for-designing-ML-pipeline.

Declaration of Generative AI and AI-Assisted Technologies

During the preparation of this work, the authors used Large Language Models (LLMs) solely for the refinement of the manuscript and for generating the TikZ code for Figure 1. All final content has been thoroughly reviewed, verified, and edited by the authors, who take full responsibility for the paper's contents.

Bibliography

[1] Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems. vol. 24 (2011)

[2] Dal Pozzolo, A., Caelen, O., Le Borgne, Y.a.l., Bontempi, G.: Credit card fraud detection (2015), https://www.kaggle.com/code/gpreda/credit-card-fraud-detection-predictive-models/input

[3] Dhaked, D.K., Narayanan, V.L.: Power output forecasting of solar photovoltaic plant using LSTM. Green Energy and Intelligent Transportation **2**(4), 100113 (2023). https://doi.org/10.1016/j.geits.2023.100113

[4] Feltes, B., et al.: Brain cancer gene expression (cumida gse50161) (2019), https://www.kaggle.com/datasets/brunogrisci/brain-cancer-gene-expression-cumida

[5] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 28 (2015), https://papers.nips.cc/paper/2015/hash/11d0e6287202fced83f79975ec59a3a6-Abstract.html

[6] Harlfoxem: King county house sales (2016), https://www.kaggle.com/datasets/harlfoxem/housesalesprediction

[7] Hutter, F., Kotthoff, L., Vanschoren, J.: Automated machine learning: methods, systems, challenges. Nature Machine Intelligence **1**(9), 423–433 (2019)

[8] Janosi, A., Steinbrunn, W., Pfisterer, M., Detrano, R.: Heart disease dataset (1988), https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

[9] Lakshmi Narayanan, V., Dhaked, D.K., Sitharthan, R.: Improved machine learning-based pitch controller for rated power generation in large-scale wind turbine. Renewable Energy Focus **50**, 100603 (2024). https://doi.org/10.1016/j.ref.2024.100603

[10] Leite, R., Brazdil, P.: Metalearning for algorithm selection: A survey. arXiv preprint arXiv:1212.5923 (2012), see also Springer survey: https://link.springer.com/article/10.1007/s10462-013-9406-y

[11] Li, L., Jamieson, K., DeSalvo, G., et al.: Hyperband: A novel bandit-based approach to hyperparameter optimization. Journal of Machine Learning Research **18**(185), 1–52 (2017)

[12] Li, P., Xi, R., Tang, J., Zhu, H.: Cleanml: A study for evaluating the impact of data cleaning on ml classification tasks. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE). pp. 13–24. IEEE (2021)

[13] Olson, R.S., Bartley, N., Urbanowicz, R.J., Moore, J.H.: TPOT: A tree-based pipeline optimization tool for automating machine learning. Journal of Machine Learning Research **17**(59), 1–5 (2016), http://jmlr.org/papers/v17/15-066.html

[14] Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD. pp. 847–855 (2013), https://arxiv.org/abs/1208.3719

[15] Vanschoren, J., Van Rijn, J.N., Bischl, B., Torgo, L.: Openml: networked science in machine learning. ACM SIGKDD Explorations Newsletter **15**(2), 49–60 (2013)

[16] Wang, C., Wu, Q., Weimer, M., Zhu, E.: FLAML: A fast and lightweight AutoML library. In: Proceedings of the 4th MLSys Conference (2021), https://arxiv.org/abs/1911.04706