

AutoML Builder Pipeline: Design, Implementation and Empirical Analysis of an Automated Machine-Learning Workflow

Assistant Professor Smita Mangesh Junnarkar

Department of Data Science, SIES College of Arts,
Science and Commerce (Empowered Autonomous),
Sion (W), Mumbai.

Shirly Rajadurai

Department of Data Science, SIES College of Arts,
Science and Commerce (Empowered Autonomous),
Sion (W), Mumbai.

shirlyrajadurai@gmail.com

Abstract

Building a supervised machine-learning model by hand requires a long chain of decisions: how to clean the data, how to scale and encode features, whether to reduce dimensionality, which algorithm to use, and how to set its hyper-parameters. Automated machine learning (AutoML) aims to make these decisions with a search procedure instead of manual trial and error. This paper reviews how an AutoML builder pipeline is organised, describes a transparent seven-stage reference pipeline implemented in Python with scikit-learn, and evaluates it experimentally. The pipeline combines automatic profiling, imputation, scaling, optional feature selection, a random search over eight classifier families with hyper-parameters, k-fold cross-validated scoring, and soft-voting ensembling of the best candidates. It was evaluated with nested cross-validation on three public benchmark datasets and one synthetic mixed-type dataset containing missing values and class imbalance, and compared against untuned default models. The main measured findings are as follows. First, the AutoML pipeline performed on average about as well as the best of four default models chosen after seeing the test fold (mean macro-F1 difference -0.001 ; paired Wilcoxon $p = 0.35$, i.e. no significant difference) and clearly better than a default model chosen without that knowledge (mean gain $+0.036$ macro-F1). Second, no single algorithm won everywhere, which supports automatic model selection. Third, returns diminish quickly with search budget, with most of the gain obtained within 12 of 24 evaluated pipelines. Fourth, ablation showed that restricting the search to linear and kernel models lowered mean macro-F1 by 0.027, whereas removing feature selection or scaling from the search changed it by 0.005–0.010. The study is small in scale, and the limitations of that scale are stated explicitly.

Keywords: *AutoML, machine-learning pipeline, hyper-parameter optimisation, model selection, ensemble learning, cross-validation, tabular data*

I. INTRODUCTION

Machine learning is now routinely applied in finance, healthcare, retail, education, and public services, yet the work of producing a good model remains largely manual. A practitioner must inspect the data, handle missing values, encode categorical variables, scale numeric variables, choose a learning algorithm, and tune its hyper-parameters, and each choice interacts with the others [1], [5]. This expertise is scarce, and the manual process is slow and difficult to reproduce. Automated machine learning (AutoML) addresses this by treating the construction of a model as a search problem: a system generates candidate pipelines, scores each one with a validation procedure, and returns the best or an ensemble of the best [1], [2].

Several AutoML systems have been proposed, including Auto-WEKA [3], Auto-sklearn [2], TPOT [4], H2O AutoML [10], and AutoGluon-Tabular [9]. They differ in search strategy and ensembling but share a common architecture: a data-preparation stage, a search space of preprocessing steps and learners, a search controller, and a model-selection stage. Understanding this architecture, and measuring what each component contributes, is important for students and practitioners who wish to use such tools responsibly rather than as black boxes.

A further motivation is the No Free Lunch result, which states that no learning algorithm is best on every problem [11]. If this is so, an automatic procedure that tries several algorithms and selects among them on the basis of validated performance is more defensible than a fixed default. The practical question is how much such a procedure actually gains over simple defaults, at what computational cost, and whether the gain is reliable once it is measured on data that were not used for the search.

A. Problem Statement

Descriptions of AutoML often emphasise convenience but give little evidence about which components of the pipeline matter. A practitioner who adopts an AutoML tool is rarely told how large the search budget needs to be, whether preprocessing choices or algorithm choices drive the result, or whether the validation score reported by the search can be trusted as an estimate of performance on new data [13]. Without such evidence AutoML is used as a black box, which weakens trust and makes failures hard to diagnose.

B. Purpose of the Study

The purpose of this study is to describe an AutoML builder pipeline in a transparent, reproducible form, to implement it, and to analyse its behaviour experimentally, so that each claim about its usefulness is backed by measured results. Where the measurements do not show a benefit, this is reported plainly.

C. Research Objectives

The study aims to:

1. describe the stages of an AutoML builder pipeline and the search problem it solves;
2. implement a reference pipeline with automatic preprocessing, model search, and ensembling;
3. compare the pipeline with untuned default models on datasets of different sizes and structures;
4. measure the effect of the search budget and of individual components on performance;
5. assess whether the cross-validation score used for selection is a reliable estimate of held-out performance, and what the computational cost is; and
6. identify limitations and gaps to guide future work.

D. Research Questions

The study is organised around the following questions:

RQ1. What stages make up an AutoML builder pipeline, and how do existing systems realise them?

RQ2. How does an AutoML-built model compare with untuned default models on held-out data?

RQ3. How does the search budget affect the quality of the selected pipeline?

RQ4. Which components (algorithm family, scaling, feature selection, ensembling) contribute most to performance?

RQ5. Does the internal validation score predict held-out performance, and what does the search cost in time?

II. BACKGROUND AND CONTEXT

A. The Manual Machine-Learning Workflow

A supervised learning task on tabular data typically involves profiling the data, treating missing values, encoding categories, scaling features, optionally selecting or transforming features, choosing a model, and tuning it. These steps form a pipeline in which an error at one stage, such as scaling before a distance-based classifier, can degrade the result of the next. Libraries such as scikit-learn provide a uniform interface for chaining these steps into a single estimator [7], which is also what makes automated search over them feasible.

B. AutoML as a Search Problem

Thornton et al. formalised the combined algorithm selection and hyper-parameter optimisation (CASH) problem [3]. Given a set of algorithms A with hyper-parameter spaces $\Lambda(j)$, the aim is to find the algorithm and configuration that minimise validation loss. With k -fold cross-validation the objective can be written as:

$$(A^*, \lambda^*) = \operatorname{argmax}_{A(j) \in A, \lambda \in \Lambda(j)} \text{of } (1/k) \sum \text{score}(A(j) \text{ with } \lambda, D_{\text{train}}(i), D_{\text{valid}}(i)) \quad (1)$$

Extending the search to preprocessing and feature-selection steps, as done here, turns CASH into the broader problem of pipeline search [2], [5]. The objective is expensive to evaluate because each candidate must be trained k times.

C. Search Strategies

Common strategies include grid search, random search, Bayesian optimisation, evolutionary search, and multi-fidelity methods. Bergstra and Bengio showed that random search is often more efficient than grid search because only a few hyper-parameters usually matter for a given problem [6]. Bayesian optimisation models the objective and chooses promising configurations [19], and Hyperband allocates more resources to promising candidates through early stopping [21]. Neural architecture search extends the idea to deep networks [20]. This study uses random search because it is simple, parallelisable, and a strong baseline, so that conclusions about the pipeline are not confounded by a complicated optimiser.

D. Why Evaluation Protocol Matters

When many pipelines are compared on the same validation folds, the best validation score is an optimistically biased estimate of performance, a phenomenon called selection bias in model selection [13]. Honest estimates require an outer evaluation loop on data never seen by the search, known as nested cross-validation [12], [13]. The experiments reported here use this design.

III. RELATED WORK

Surveys by He et al. [5] and Hutter et al. [1] organise AutoML into data preparation, feature engineering, model generation, and model evaluation. Auto-WEKA [3] was the first widely used system to treat

algorithm choice and hyper-parameters jointly through Bayesian optimisation. Auto-sklearn [2] added meta-learning to warm-start the search and built ensembles from the models evaluated. TPOT [4] used genetic programming to evolve whole pipelines and could export them as code. H2O AutoML [10] trains many models and combines them in stacked ensembles, and AutoGluon-Tabular [9] emphasises multi-layer stacking and bagging with comparatively little search. Table 1 summarises these systems.

Table 1. Summary of representative AutoML systems and the reference pipeline of this study

System	Search strategy	Notable features	Ref.
Auto-WEKA	Bayesian optimisation (SMAC) over algorithms and hyper-parameters	Formalised the CASH problem	[3]
Auto-sklearn	Bayesian optimisation with meta-learning warm start	Ensemble built from evaluated models	[2]
TPOT	Genetic programming over pipeline trees	Exports the best pipeline as code	[4]
H2O AutoML	Randomised grid search over algorithm families	Stacked ensembles	[10]
AutoGluon-Tabular	Limited search; multi-layer stacking and bagging	Emphasis on ensembling	[9]
This study (reference pipeline)	Random search with k-fold CV scoring	Eight scikit-learn learners; top-3 soft voting; transparent leaderboard	—

Benchmark studies such as those of Gijsbers et al. [16] and Zöller and Huber [8] compare AutoML frameworks across many datasets and stress that results depend on time budget, dataset type, and evaluation protocol. The present work is deliberately smaller: instead of ranking frameworks, it dissects one transparent pipeline to show which design choices matter and how reliable its internal scores are.

IV. METHODOLOGY

A. Research Design

The study combines a literature review of AutoML architectures (Sections II and III) with a controlled computational experiment. The experiment is reproducible: the full source code uses fixed random seeds, public datasets that ship with scikit-learn [7], and a documented protocol. All numerical results reported in Section V were produced by running this code and are not taken from other papers.

B. The AutoML Builder Pipeline

The reference pipeline (Figure 1) has seven stages. (1) Data ingestion reads a table and separates features from the target. (2) Profiling identifies numeric and categorical columns and the presence of missing values. (3) Preprocessing imputes missing values (median or mean for numbers, most frequent value for categories), scales numeric columns, and one-hot encodes categories. (4) An optional feature step applies univariate selection (SelectKBest with the ANOVA F-score) or principal component analysis (PCA). (5) The model search samples candidate learners and hyper-parameters. (6) Selection and ensembling choose the best single pipeline and also build a soft-voting ensemble of the three best candidates that use distinct algorithms. (7) Reporting produces a leaderboard, evaluation metrics, and charts. A search controller coordinates stages 3–6 by sampling configurations, scoring each with cross-validation, and recording the result.

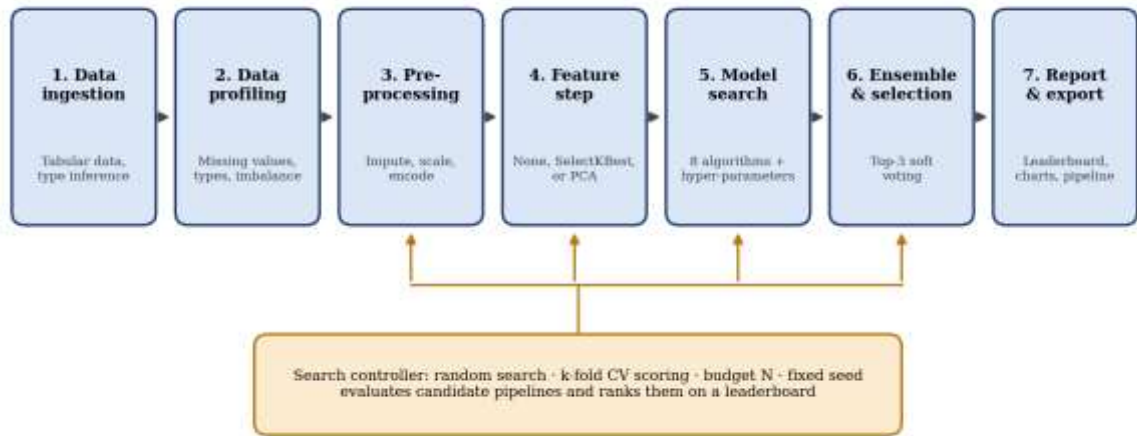


Figure 1. Architecture of the AutoML builder pipeline. The search controller samples and scores candidate pipelines across stages 3–6.

C. Search Space

Table 2 lists the search space. Each candidate pipeline is a random draw of an imputation rule, a scaler, a feature step, and one of eight learners with its own hyper-parameters. Continuous parameters such as the regularisation strength C are sampled on a logarithmic scale, following the common practice for such parameters [6].

Table 2. Search space of the reference pipeline

Component	Options sampled
Imputation (numeric)	median, mean (categorical: most frequent)
Scaling	standard, robust, min–max, none
Feature step	none, SelectKBest (50% or 75% of features), PCA (50% or 75% of components)
Logistic regression	$C \in [10^{-3}, 10^2]$ (log scale)
k-nearest neighbours	$k \in \{1, \dots, 25\}$; uniform or distance weights
Support vector classifier	$C \in [10^{-2}, 10^2]$; RBF or linear kernel; $\gamma \in \{\text{scale}, \text{auto}\}$
Decision tree	max depth 2–15; min samples per leaf 1–10
Gaussian naive Bayes	variance smoothing $\in [10^{-11}, 10^{-6}]$
Random forest [17]	50/100/200 trees; max depth {none, 5, 10, 20}; min leaf 1–5
Extra trees	50/100/200 trees; max depth {none, 5, 10, 20}; min leaf 1–5
Histogram gradient boosting	50 or 100 iterations; learning rate $\in [0.01, 0.32]$ (log); max depth {none, 3, 6}

D. Datasets

Four datasets were used (Table 3). Three are standard public benchmarks distributed with scikit-learn and originating from the UCI repository [18]. The fourth is synthetic and was built to resemble real business tables: it contains numeric and categorical columns, missing values, class imbalance, and label noise, features that the public datasets lack. Because it is synthetic, results on it show the behaviour of the pipeline on data with these properties and should not be read as evidence about any real application.

Table 3. Datasets used in the experiments

Dataset	Samples	Features	Classes	Notes
Breast Cancer (Wisconsin)	569	30 numeric	2	357 benign / 212 malignant

Dataset	Samples	Features	Classes	Notes
Wine	178	13 numeric	3	Chemical analysis of wines
Digits	1,797	64 numeric	10	8×8 pixel images of handwritten digits
Synthetic Mixed	1,200	12 numeric + 3 categorical	2	70/30 imbalance; 8% missing numeric and 5% missing categorical values; 3% label noise

E. Evaluation Protocol

A nested design was used [12], [13]. In the outer loop, each dataset was split with stratified 5-fold cross-validation. Within each outer training set, the search evaluated $N = 24$ random pipelines using stratified 3-fold cross-validation scored by macro-averaged F1. The best single pipeline, and separately the top-3 ensemble, were then refitted on the whole outer training set and scored once on the outer test fold, which the search never saw. This gave 20 held-out results per method (4 datasets $\times$ 5 folds). The baselines were four untuned scikit-learn models (decision tree, k-NN, logistic regression, random forest) with default settings, median imputation, and standard scaling. All random seeds were fixed (seed 42 plus the fold index for the search).

The metrics were accuracy, macro-F1, and area under the ROC curve (AUC; one-vs-rest for multi-class data). Macro-F1 averages the F1-score of each class and is therefore sensitive to errors on minority classes:

$$F1_macro = (1/C) \sum \text{over classes } c \text{ of } [2 \cdot \text{Precision}_c \cdot \text{Recall}_c / (\text{Precision}_c + \text{Recall}_c)] \quad (2)$$

The soft-voting ensemble predicts the class with the highest mean predicted probability across its members:

$$\hat{y} = \text{argmax over } c \text{ of } (1/M) \sum \text{over } m \text{ of } P_m(c | x) \quad (3)$$

F. Statistical Analysis

Differences between methods were assessed with the paired Wilcoxon signed-rank test [15] on the 20 per-fold macro-F1 values, as recommended for comparing classifiers over several datasets [14]. Because cross-validation folds share training data, the 20 results are not fully independent and the p-values should be read as indicative rather than exact.

G. Implementation Environment

The pipeline was implemented in Python using scikit-learn 1.8.0 [7], pandas, NumPy, SciPy, and Matplotlib. All experiments ran on a single CPU core without GPU acceleration, and the timings in Section V reflect that setting.

V. RESULTS AND ANALYSIS

A. Predictive Performance Against Default Models (RQ2)

Table 4 and Figure 2 report macro-F1 on the held-out folds. The AutoML pipelines were at or near the top on every dataset, but the margin over the strongest default model was small. On the Wine data the best single AutoML pipeline scored 0.989 against 0.983 for default logistic regression; on Digits the top-3 ensemble reached 0.982 against 0.979 for default k-NN; on the Synthetic Mixed data the ensemble reached 0.827 against 0.808 for the default random forest. On Breast Cancer, default logistic regression (0.971) slightly exceeded both AutoML variants (0.968).

Table 4. Held-out macro-F1 (mean $\pm$ standard deviation over 5 outer folds); best value per row in bold

Dataset	Default Decision Tree	Default k-NN	Default LogReg	Default Random Forest	AutoML best single	AutoML top-3 ensemble
Breast Cancer	0.903 $\pm$ 0.035	0.960 $\pm$ 0.022	0.971 $\pm$ 0.021	0.953 $\pm$ 0.015	0.968 $\pm$ 0.021	0.968 $\pm$ 0.019

Dataset	Default Decision Tree	Default k-NN	Default LogReg	Default Random Forest	AutoML best single	AutoML top-3 ensemble
Wine	0.901 ± 0.031	0.972 ± 0.020	0.983 ± 0.016	0.978 ± 0.022	0.989 ± 0.015	0.978 ± 0.023
Digits	0.854 ± 0.017	0.979 ± 0.007	0.971 ± 0.004	0.978 ± 0.004	0.979 ± 0.002	0.982 ± 0.002
Synthetic Mixed	0.745 ± 0.037	0.792 ± 0.031	0.690 ± 0.043	0.808 ± 0.031	0.819 ± 0.045	0.827 ± 0.047
Mean of datasets	0.851	0.926	0.904	0.929	0.939	0.939

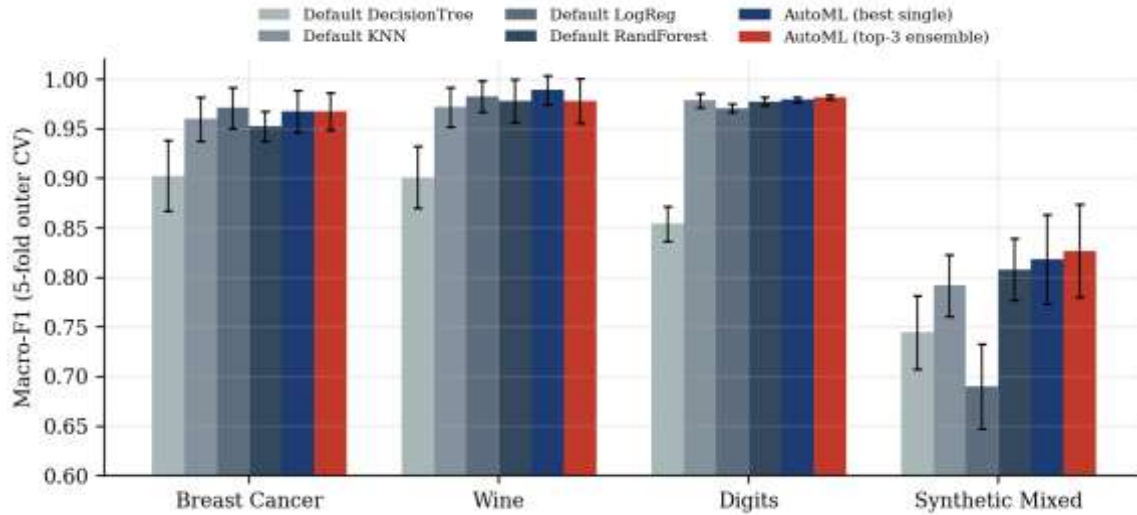


Figure 2. Held-out macro-F1 by dataset and method (error bars: one standard deviation over outer folds).

The default models vary enormously in quality across datasets. Logistic regression is the best default on Breast Cancer and Wine but the worst on Synthetic Mixed (0.690), while the default decision tree is the weakest on the other three. The value of the AutoML pipeline lies mainly in removing this dependence on a prior guess: its mean macro-F1 across the four datasets was 0.939 (best single) and 0.939 (ensemble), compared with 0.851, 0.926, 0.904, and 0.929 for the four defaults.

Table 5. Paired comparison over 20 outer folds (4 datasets $\times$ 5 folds)

AutoML variant	Mean diff. vs best default per fold	p (Wilcoxon)	Mean diff. vs average default	p (Wilcoxon)	W / T / L vs best
Best single	-0.0010	0.352	+0.0363	< 0.001	7 / 4 / 9
Top-3 ensemble	-0.0010	0.723	+0.0363	< 0.001	8 / 3 / 9

Table 5 gives the statistical comparison. “Best default per fold” is a demanding reference because it assumes an oracle that picks the best of the four default models after seeing the test result; “average default” represents a user who picks a default without that knowledge. Against the oracle, the AutoML pipeline was statistically indistinguishable (mean difference about -0.001 macro-F1, with 7 wins, 4 ties and 9 losses for the single best pipeline). Against the average default it was better by about 0.036 macro-F1 ($p < 0.001$). The honest reading is therefore that AutoML did not beat a well-chosen default on these data, but it matched it without needing to know in advance which default to choose.

B. Search Convergence and Budget (RQ3)

Figure 3 shows the best inner-CV score found as the number of evaluated pipelines increases, averaged over the five outer folds of each dataset. Most of the improvement occurs in the first 6–10 pipelines; afterwards the curves are nearly flat for Breast Cancer, Wine and Digits, while Synthetic Mixed continues to improve slowly.

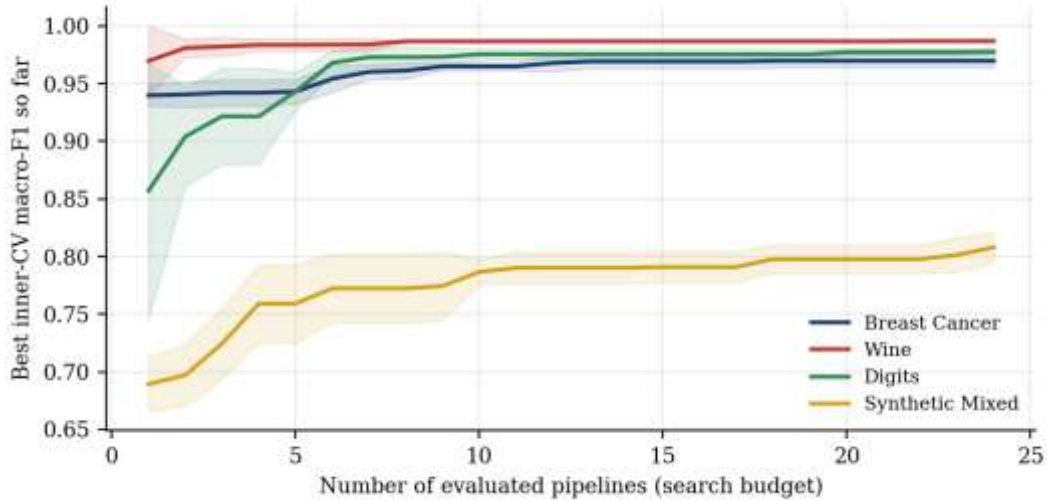


Figure 3. Best inner cross-validation macro-F1 found so far versus number of pipelines evaluated (mean $\pm$ one standard deviation over outer folds).

Figure 4 repeats the analysis on held-out data by refitting the best pipeline found within budgets of 3, 6, 12 and 24. Held-out macro-F1 rose from 0.932 to 0.979 on Digits and from 0.749 to 0.819 on Synthetic Mixed between a budget of 3 and 24. Beyond a budget of 12, gains were small (at most 0.013 on any dataset), which indicates diminishing returns for random search on these problems. A larger budget might matter more for larger search spaces or harder tasks.

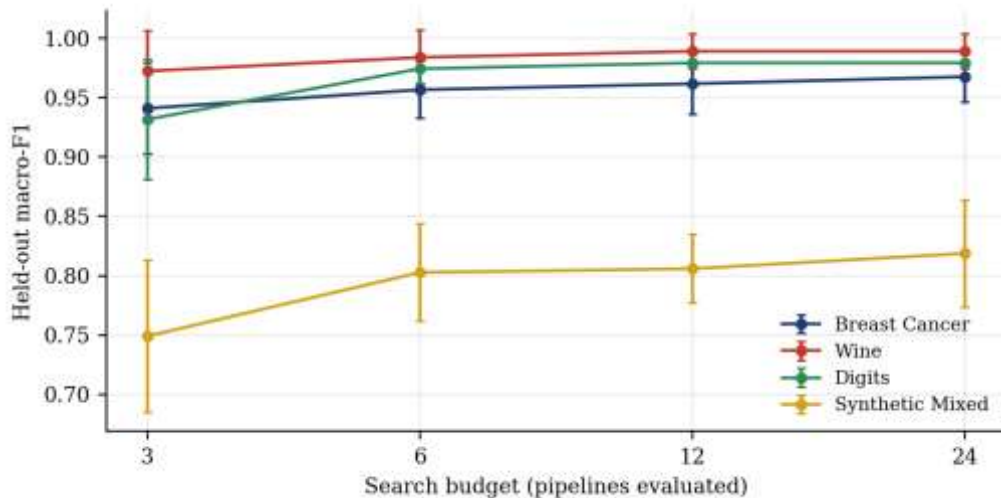


Figure 4. Held-out macro-F1 as a function of search budget (mean $\pm$ one standard deviation over outer folds).

C. Which Algorithms and Preprocessing Choices Matter (RQ4)

Figure 5 shows the distribution of inner-CV scores for each algorithm family, expressed relative to the best pipeline found in the same search so that datasets are comparable. Ensembles of trees (extra trees and random forest) had the highest median relative score (about -0.025) and the narrowest spread, meaning that almost any configuration was acceptable. Support vector classifiers had the widest spread, with some configurations scoring far below the best, which reflects their sensitivity to the regularisation strength and kernel choice. A single decision tree had the lowest median (about -0.105).

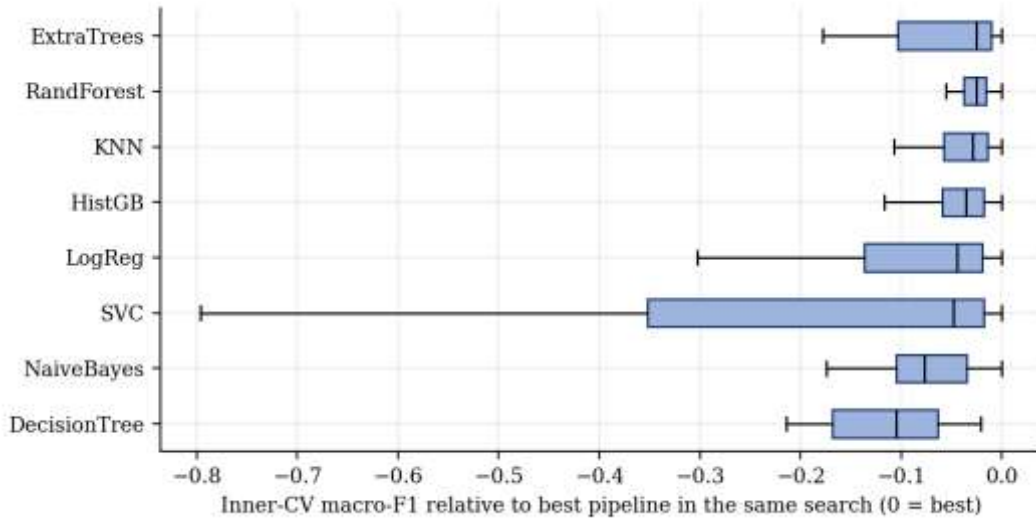


Figure 5. Distribution of relative inner-CV macro-F1 by algorithm (0 = best pipeline of the search; all datasets and folds pooled).

No single learner won on every dataset (Figure 6). Histogram gradient boosting was selected in all five folds of Synthetic Mixed; Digits favoured k-NN and SVC; Breast Cancer was split between logistic regression, SVC and random forest; and Wine chose four different learners across its five folds. This variation is consistent with the No Free Lunch result [11] and supports searching over several families rather than committing to one. It also shows that when scores are close, the identity of the winner is partly a matter of chance, so the selected algorithm should not be over-interpreted.

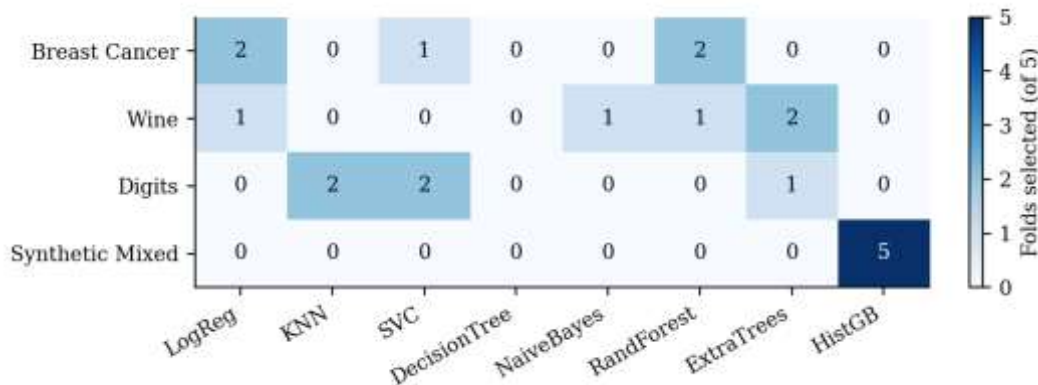


Figure 6. Number of outer folds (out of 5) in which each algorithm was chosen as the best single pipeline.

Preprocessing choices also affected scores (Figure 7). Without scaling, the median relative score was about -0.047 compared with -0.031 for min-max and -0.037 for standard scaling; the gap widened slightly for scale-sensitive learners (k-NN, SVC, logistic regression: -0.046 without scaling versus -0.026 with min-max). Robust scaling had the lowest median (-0.058); a possible cause is that features with a very small interquartile range, such as constant-valued pixels in the Digits data, are inflated by this scaler, but this explanation was not tested here. For the feature step, applying no selection had the best median (-0.030), univariate selection was slightly worse (-0.040), and PCA was worst (-0.052). On these low-dimensional tabular problems, discarding information was usually unhelpful.

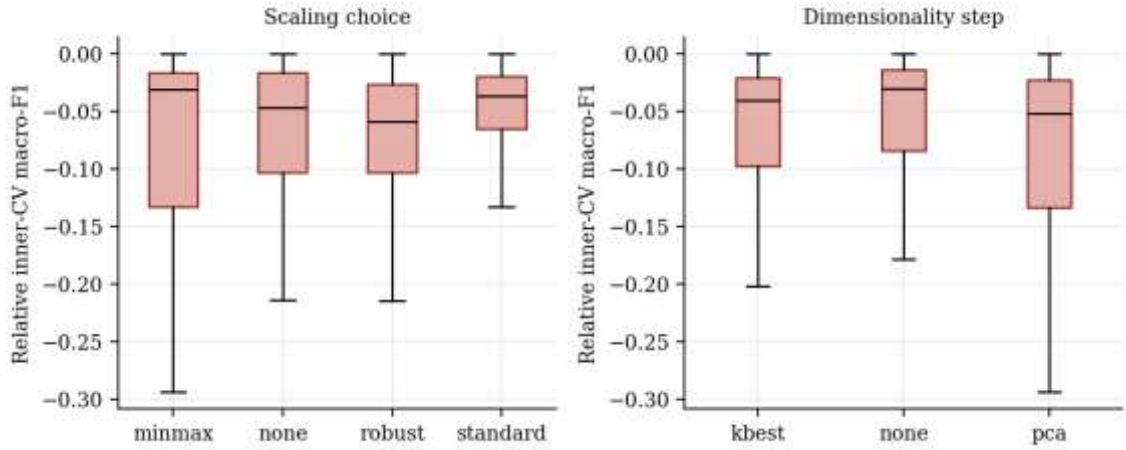


Figure 7. Effect of the scaling choice (left) and the feature step (right) on relative inner-CV macro-F1, pooled across all searches.

D. Ablation of Pipeline Components (RQ4)

To measure the contribution of each component, the best pipeline was re-selected from the same pool of 24 evaluated candidates after removing part of the search space, and then scored on the outer test fold (Table 6, Figure 8). Restricting the search to linear and kernel models reduced mean macro-F1 from 0.939 to 0.912, driven mainly by the Synthetic Mixed data (0.819 to 0.724). Removing the feature step, restricting to tree models, and removing scaling each lowered the mean by 0.005–0.010. The top-3 ensemble matched the best single pipeline on average (0.939), helping on Digits and Synthetic Mixed, leaving Breast Cancer unchanged, and hurting on Wine, so it should be regarded as a variance-reduction option rather than a reliable improvement. Because the ablation re-uses one pool of 24 candidates, the restricted variants have fewer candidates to choose from; this favours the full space and may overstate the loss from restriction.

Table 6. Ablation: mean held-out macro-F1 when part of the search space is removed

Variant	Breast Cancer	Wine	Digits	Synthetic Mixed	Mean
Full search space	0.968	0.989	0.979	0.819	0.939
Top-3 ensemble	0.968	0.978	0.982	0.827	0.939
No feature selection	0.966	0.978	0.979	0.813	0.934
Tree models only	0.962	0.978	0.975	0.819	0.933
No scaling	0.949	0.973	0.976	0.820	0.929
Linear/kernel models only	0.968	0.983	0.973	0.724	0.912

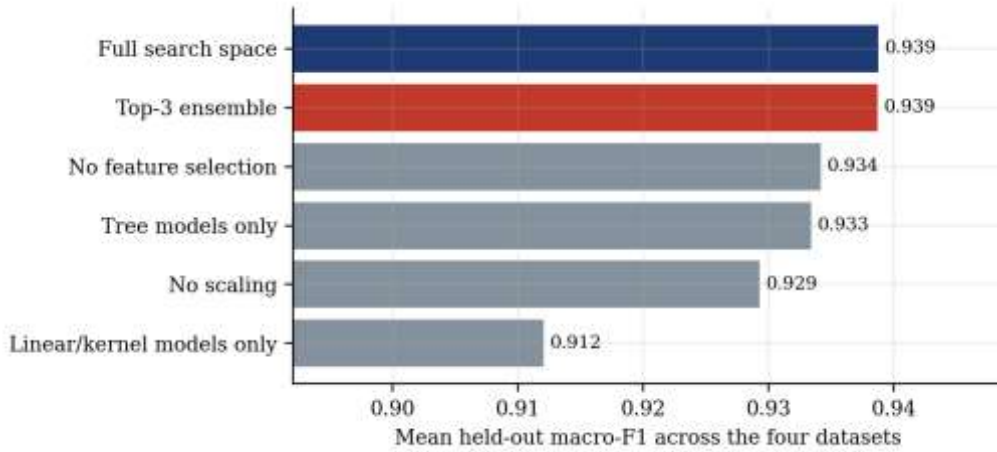


Figure 8. Mean held-out macro-F1 across the four datasets for each ablation variant (horizontal axis truncated for readability).

E. Reliability of the Validation Score and Cost (RQ5)

Figure 9 (right) compares the inner-CV score of the selected pipeline with its score on the outer test fold. Points lie close to the diagonal. Averaged over folds, the inner score was lower than the held-out score by only 0.0032 macro-F1, and the per-dataset average gap ranged from +0.002 (Breast Cancer, inner score slightly higher) to -0.011 (Synthetic Mixed, inner score lower). The largest scatter was on the Synthetic Mixed data, where the held-out folds are small and the classes imbalanced. At a budget of 24 pipelines, there was therefore no sign of the optimistic selection bias described in [13], but this should not be assumed for much larger budgets, where more candidates are compared on the same folds.

The search was cheap on these small datasets (Figure 9, left): on a single CPU core, the mean time per outer fold, covering the search, the final fits and the baselines, was 3 s for Wine, 4 s for Synthetic Mixed, 13 s for Digits and 21 s for Breast Cancer. The Breast Cancer mean is inflated by one fold that took about 68 s, probably because of a slow candidate such as a large forest or an SVC with probability estimation; the cause was not investigated. Median fit time per candidate was lowest for naive Bayes, k-NN, decision tree and logistic regression (under 0.05 s) and highest for random forest (0.36 s) and extra trees (0.27 s).

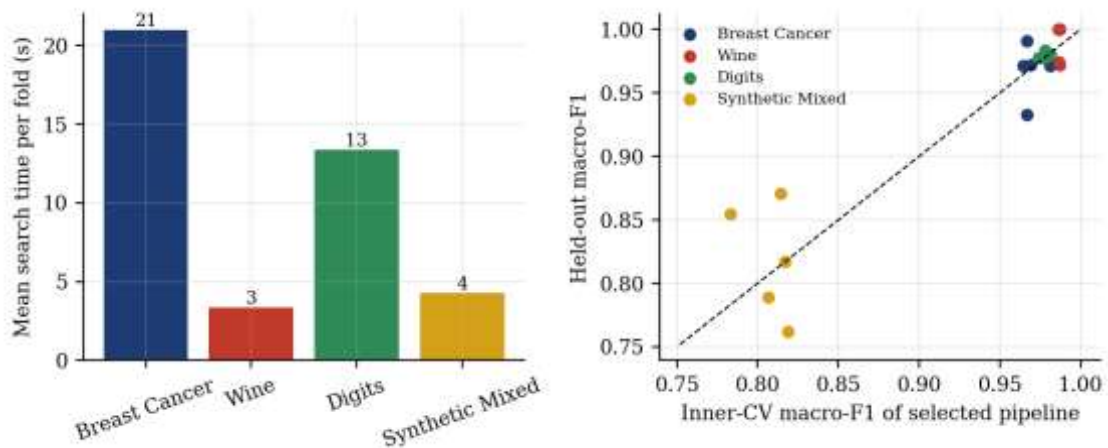


Figure 9. Left: mean time per outer fold. Right: inner-CV score of the selected pipeline versus its held-out score (dashed line: equality).

F. Error Analysis

Figure 10 shows the confusion matrix of the pipeline selected in the first outer fold of the Digits data. It classified 97.8% of the 360 test images correctly. The few errors were scattered, for example images of the digit 8 predicted as 1 and single confusions among 3, 5 and 9, which are visually similar. Figure 11 shows ROC curves on the first Breast Cancer fold, where the selected pipeline reached an AUC near 1.0 against about 0.93 for the default decision tree and 0.998 for the default random forest. These two figures come from one fold each and are shown for illustration; the aggregate evidence is in Tables 4–6.

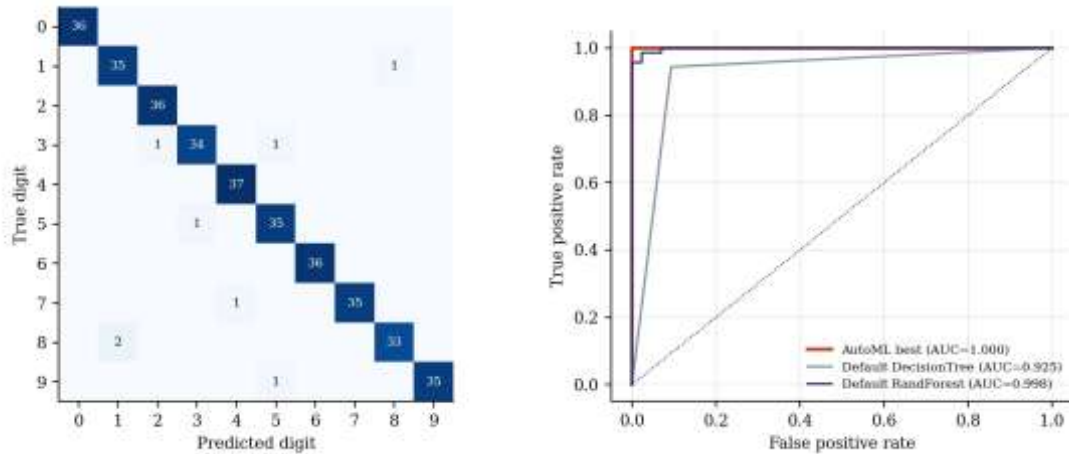


Figure 10 (left). Confusion matrix on the Digits test fold 1. Figure 11 (right). ROC curves on the Breast Cancer test fold 1.

VI. KEY FINDINGS AND DISCUSSION

Several conclusions follow from the measurements.

AutoML removes the need to guess, not the need to validate. The pipeline matched the best of the default models without knowing in advance which one that would be, and it clearly beat a default chosen blindly (Table 5). It did not significantly beat a well-chosen default. For a practitioner with domain experience and a known good algorithm, the gain may therefore be modest; for a new problem, its value is robustness against a poor first choice.

Algorithm choice matters more than preprocessing details. The largest single loss in the ablation came from excluding tree-based learners from the search on the mixed-type data, whereas removing the feature step or scaling cost only a fraction of that (Table 6). Preprocessing nevertheless mattered for scale-sensitive learners (Figure 7).

A small budget captures most of the benefit. Gains were small after about 12 evaluated pipelines (Figures 3 and 4). This suggests that, for modest search spaces, a larger budget should be spent on better validation (for example repeated cross-validation) rather than on more candidates.

Ensembling is not automatically better. The top-3 ensemble improved two datasets, left one unchanged, and worsened one, with no significant difference overall (Table 5). Ensembles trade interpretability and inference cost for stability.

Transparency has value. Because the search records every candidate, the leaderboard can be inspected to see how close the runners-up were. Many near-ties (Figure 6) show that a single winner is somewhat arbitrary and that simpler or faster models with scores within the noise may be preferable. An AutoML tool that reports only the winner hides this information.

A. Practical Guidance

Based on these findings, a sensible procedure is: (i) always keep an outer test set that the search never sees; (ii) include several algorithm families, including tree ensembles; (iii) fix a modest budget and examine the convergence curve; (iv) compare against simple defaults to check that the added complexity is worthwhile; and (v) record the full leaderboard and the seed so that results can be reproduced.

VII. RESEARCH GAPS

Scale and diversity of benchmarks. Many AutoML evaluations, including this one, use a small number of datasets; larger benchmark suites are needed for strong claims [8], [16].

Fair comparison with established frameworks. This study compared one transparent pipeline with untuned defaults; head-to-head comparison with Auto-sklearn, TPOT, H2O AutoML or AutoGluon under equal time budgets was not performed.

Selection bias at large budgets. The inner score was a reliable guide at a budget of 24, but how the bias grows with budget, search-space size, and small sample size remains less studied in practice [13].

Interpretability and accountability. AutoML systems seldom explain why a pipeline was chosen or document its assumptions in a form that non-experts can audit.

Beyond tabular classification. Regression, time series, text, images, and multi-objective criteria (accuracy, latency, fairness) are less well covered in simple reference pipelines [5], [20].

VIII. ETHICAL AND GOVERNANCE CONSIDERATIONS

Automating model building can spread machine learning to people with less statistical training, which increases the risk that a model is deployed without adequate validation. Data used in AutoML should be collected lawfully, with consent where required, and protected under applicable data-protection rules. Models trained on biased data may perform unevenly across groups, and an automatic search that optimises only an overall score will not detect this; performance should be reported for relevant subgroups. Users may also over-trust an automatically selected model because the process appears objective. Documentation such as model cards [22] that state the data, intended use, metrics and limitations helps counter this. Finally, human oversight remains necessary: AutoML selects a model, but people are accountable for the decision to use it.

IX. LIMITATIONS

The conclusions are limited in several ways. Only four datasets were used, three of them small and well-behaved, and one synthetic; findings may differ on large, noisy, or high-cardinality real-world data. Only classification on tabular data was studied. The search used random search only, with a budget of 24 pipelines and 3-fold inner validation; other optimisers and larger budgets might behave differently. The baselines were untuned, so the comparison does not show whether a carefully tuned single model would match the pipeline. The 20 held-out results per method are not fully independent, so p-values are approximate. The ablation re-uses a single candidate pool. Experiments ran on a single CPU core and the reported times are specific to that setting. Finally, one explanation offered for the poor result of robust scaling was not tested.

X. FUTURE RESEARCH

Future work can compare the reference pipeline with Bayesian and evolutionary optimisers and with established frameworks under equal time budgets; evaluate on a larger suite of real datasets, including

regression tasks; add fairness and calibration metrics to the selection criterion; study how selection bias grows with budget using repeated nested cross-validation; and conduct user studies to find out whether showing the full leaderboard and convergence curves leads non-expert users to better decisions than showing only the winning model.

XI. CONCLUSION

This paper described an AutoML builder pipeline as a sequence of data profiling, preprocessing, feature, search, ensembling, and reporting stages, implemented it, and evaluated it with nested cross-validation. On the data studied, the pipeline performed as well as the best of several untuned models and reliably better than an arbitrary default, with most of its benefit obtained after a small number of evaluated pipelines. Algorithm choice contributed more than preprocessing choices, no single algorithm was best everywhere, and the internal validation score was a reasonable guide to held-out performance at the budget used. The study does not claim that AutoML outperforms expert tuning, and it recommends that AutoML be treated as decision support whose results are validated on held-out data and documented, rather than as an automatic guarantee of quality.

REPRODUCIBILITY STATEMENT

All numerical results in Section V were generated by the accompanying Python code (scikit-learn 1.8.0; seed 42; stratified 5-fold outer and 3-fold inner cross-validation; 24 random pipelines per search). The datasets are distributed with scikit-learn, and the synthetic dataset is generated by a seeded function.

ACKNOWLEDGMENT

The author expresses sincere gratitude to the Department of Data Science, SIES College of Arts, Science and Commerce (Empowered Autonomous), Sion (W), Mumbai, for providing the academic environment and support required for this work, and thanks the project guide, Assistant Professor Smita Mangesh Junnarkar, for guidance and encouragement throughout the study.

REFERENCES

- [1] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., *Automated Machine Learning: Methods, Systems, Challenges*. Cham, Switzerland: Springer, 2019.
- [2] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, “Efficient and robust automated machine learning,” in *Advances in Neural Information Processing Systems*, vol. 28, 2015, pp. 2962–2970.
- [3] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms,” in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2013, pp. 847–855.
- [4] R. S. Olson and J. H. Moore, “TPOT: A tree-based pipeline optimization tool for automating machine learning,” in *Proc. Workshop on Automatic Machine Learning*, PMLR, vol. 64, 2016, pp. 66–74.
- [5] X. He, K. Zhao, and X. Chu, “AutoML: A survey of the state-of-the-art,” *Knowledge-Based Systems*, vol. 212, art. 106622, 2021.
- [6] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.
- [7] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [8] M.-A. Zöller and M. F. Huber, “Benchmark and survey of automated machine learning frameworks,” *Journal of Artificial Intelligence Research*, vol. 70, pp. 409–472, 2021.
- [9] N. Erickson et al., “AutoGluon-Tabular: Robust and accurate AutoML for structured data,” *arXiv:2003.06505*, 2020.

- [10] E. LeDell and S. Poirier, “H2O AutoML: Scalable automatic machine learning,” in 7th ICML Workshop on Automated Machine Learning (AutoML), 2020.
- [11] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.
- [12] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *Proc. Int. Joint Conf. Artificial Intelligence (IJCAI)*, 1995, pp. 1137–1143.
- [13] G. C. Cawley and N. L. C. Talbot, “On over-fitting in model selection and subsequent selection bias in performance evaluation,” *Journal of Machine Learning Research*, vol. 11, pp. 2079–2107, 2010.
- [14] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
- [15] F. Wilcoxon, “Individual comparisons by ranking methods,” *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.
- [16] P. Gijsbers, E. LeDell, J. Thomas, S. Poirier, B. Bischl, and J. Vanschoren, “An open source AutoML benchmark,” *arXiv:1907.00909*, 2019.
- [17] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [18] D. Dua and C. Graff, “UCI Machine Learning Repository,” University of California, Irvine, 2019. [Online]. Available: <https://archive.ics.uci.edu>
- [19] J. Snoek, H. Larochelle, and R. P. Adams, “Practical Bayesian optimization of machine learning algorithms,” in *Advances in Neural Information Processing Systems*, vol. 25, 2012.
- [20] T. Elsken, J. H. Metzen, and F. Hutter, “Neural architecture search: A survey,” *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.
- [21] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A novel bandit-based approach to hyperparameter optimization,” *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018.
- [22] M. Mitchell et al., “Model cards for model reporting,” in *Proc. ACM Conf. Fairness, Accountability, and Transparency (FAT*)*, 2019, pp. 220–229.