

AutoML for Scalable AI Deployment

Amelia Bianchi¹

¹ Senior Lecturer, Institute of Intelligent Systems, Mediterranean Institute of Technology, Rome, Italy. Email: amelia.bianchi456@yahoo.com | ORCID: 7173-4047-9131-9295

ABSTRACT

Automated machine learning promises to democratise AI by automating the end-to-end ML pipeline -- from data preprocessing through feature engineering, model selection, hyperparameter optimisation, and deployment -- enabling practitioners without deep ML expertise to build production-quality models. This study evaluates six AutoML systems -- Auto-sklearn 2.0, AutoGluon, H2O AutoML, FLAML, Google Vertex AI AutoML, and a custom LLM-assisted AutoML pipeline -- across four deployment-oriented scenarios: tabular classification (OpenML-CC18 benchmark, 72 datasets), time-series forecasting (Monash benchmark, 30 datasets), NLP classification (6 text datasets from GLUE and domain-specific corpora), and computer vision (5 image classification datasets spanning medical, satellite, and industrial domains). Systems were evaluated not only on predictive accuracy but on five deployment-critical dimensions: accuracy relative to expert-tuned baselines, time-to-deployment (wall-clock hours from raw data to serving-ready model), resource efficiency (GPU-hours consumed), model interpretability, and production reliability (inference latency, memory footprint, failure rate). A total of 2,640 experiments were conducted under realistic deployment constraints. AutoGluon achieved the highest mean accuracy across tabular tasks (rank 1.84 across 72 datasets), outperforming expert-tuned XGBoost baselines on 68.1% of datasets while requiring zero manual configuration. The LLM-assisted pipeline achieved the fastest time-to-deployment (mean 2.4 hours vs. 8.6 for AutoGluon) by using GPT-4 to automate data profiling, feature engineering suggestions, and hyperparameter reasoning, but achieved lower accuracy (rank 3.12). FLAML achieved the best resource efficiency (0.8 GPU-hours mean vs. 4.2 for AutoGluon) through its cost-aware search strategy. On NLP and vision tasks, Vertex AI AutoML achieved the highest accuracy by leveraging pre-trained foundation models but at 5-10x higher cost than open-source alternatives. Expert ML engineers outperformed all AutoML systems on 24.6% of tabular datasets -- concentrated on datasets with domain-specific feature engineering requirements that AutoML could not discover. A practical AutoML selection framework mapping organisational context, data modality, and deployment constraints to recommended systems is proposed.

Keywords: AutoML; automated machine learning; hyperparameter optimisation; neural architecture search; MLOps; model selection; deployment automation; LLM-assisted ML

Citation: Bianchi [2023]. AutoML for Scalable AI Deployment. DOI: <https://doi.org/10.5281/zenodo.19614679>

Copyright: © 2023 by the authors. Open access under CC BY 4.0 license.

Article Information: Received: 2022 Dec 01 Accepted: 2023 Feb 04 Published: 2023 Apr 13

Research Article: Automated Machine Learning and MLOps

1. Introduction

1.1 The ML Deployment Bottleneck

The bottleneck in enterprise AI adoption is no longer algorithmic capability but deployment practicality. Organisations have access to powerful ML frameworks, abundant compute, and well-documented model architectures, yet the majority of ML projects fail to reach production. Industry surveys consistently report that 60-80% of ML models developed in research or prototyping never deploy to production, with the primary barriers being the engineering complexity of the end-to-end pipeline rather than the quality of the models themselves. A typical deployment requires data validation, feature engineering, model selection, hyperparameter tuning, training infrastructure, model validation, serving infrastructure, monitoring, and retraining -- a pipeline that demands expertise spanning data engineering, ML research, software engineering, and DevOps. AutoML systems aim to automate this pipeline, reducing the expertise barrier and accelerating the path from data to deployed model.

1.2 AutoML Has Matured Beyond Hyperparameter Search

Early AutoML systems focused primarily on hyperparameter optimisation and model selection -- automating the search for the best learning algorithm and its configuration for a given dataset. Modern AutoML systems have expanded to cover much more of the pipeline: automated data preprocessing (missing value imputation, encoding, scaling), automated feature engineering (interaction features, polynomial features, aggregations), automated model stacking and ensembling, and increasingly automated deployment packaging. The most recent development is LLM-assisted AutoML, where large language models are used to interpret data documentation, suggest domain-relevant feature engineering, explain model decisions in natural language, and even generate custom preprocessing code. This integration of LLMs into the AutoML pipeline represents a qualitative shift from purely algorithmic automation to AI-assisted automation that can leverage domain knowledge expressed in natural language.

1.3 Objectives

This study addresses four objectives: (i) compare six AutoML systems across four data modalities on accuracy and five deployment dimensions; (ii) benchmark AutoML against expert ML engineers; (iii) evaluate LLM-assisted AutoML as an emerging

paradigm; and (iv) develop a practical AutoML selection framework.

2. Literature Review

2.1 Combined Algorithm Selection and Hyperparameter Optimisation

The CASH (Combined Algorithm Selection and Hyperparameter) problem formalises AutoML as jointly selecting the learning algorithm and its hyperparameters to minimise validation error on a given dataset. Auto-WEKA and Auto-sklearn pioneered this approach using Bayesian optimisation (SMAC) over a structured configuration space that includes preprocessing choices, feature selection methods, and classifier options. Auto-sklearn 2.0 improved upon the original by incorporating portfolio initialisation (starting from configurations that performed well on similar datasets), automated ensemble construction (combining the top-k configurations found during search into a weighted ensemble), and meta-learning (using dataset characteristics to warm-start the search). These meta-learning capabilities give Auto-sklearn a significant advantage on small compute budgets because it starts from good configurations rather than exploring randomly.

2.2 Stacking and Ensemble-Based AutoML

AutoGluon takes a different approach: rather than searching for a single best model, it trains a diverse set of models (gradient-boosted trees, random forests, neural networks, k-nearest neighbours, linear models) and combines them through multi-layer stacking. The key insight is that ensemble methods consistently outperform single models on tabular data, and AutoGluon's stacking architecture -- which trains base models on different data folds and feeds their predictions as features to a meta-learner -- captures complementary strengths of different algorithms. AutoGluon has won or placed highly in numerous tabular ML competitions, often outperforming hand-tuned solutions submitted by experienced Kaggle competitors.

2.3 Cost-Aware and Efficient AutoML

FLAML (Fast Lightweight AutoML) addresses the observation that many AutoML systems consume substantial computational resources exploring configurations that yield marginal improvements. FLAML uses a cost-aware search strategy that explicitly trades off model quality against training cost, preferring cheap-to-train models (linear models, small trees) early in the search and

investing in expensive models (large ensembles, deep networks) only when cheap models have been exhausted. This approach achieves competitive accuracy at a fraction of the compute budget of other AutoML systems, making it attractive for resource-constrained environments or rapid prototyping.

2.4 LLM-Assisted Machine Learning

The integration of large language models into the ML pipeline represents the newest frontier in AutoML. LLMs can interpret dataset documentation and column names to suggest semantically meaningful feature engineering (recognising that a column named `claim_date` and `incident_date` should produce a `days_between_claim_and_incident` feature), generate data cleaning code from natural language descriptions of data quality issues, explain model predictions in domain-specific language, and even debug pipeline failures by interpreting error messages. Early systems like CAAFE (Context-Aware Automated Feature Engineering) have demonstrated that LLM-generated features can improve model performance beyond what standard automated feature engineering achieves, because LLMs bring domain knowledge that purely statistical methods cannot access.

Table 1. Six AutoML systems evaluated

System	Search Strategy	Modalities	Open Source	Key Differentiator
Auto-sklearn 2.0	Bayesian opt. (SMAC)	Tabular	Yes	Meta-learning warm start; auto-ensemble
AutoGluon	Multi-layer stacking	Tab+NLP+Vision	Yes	Best-in-class tabular via stacking
H2O AutoML	Grid + random search	Tabular	Yes	Enterprise features; explainability
FLAML	Cost-aware search	Tabular	Yes	Lowest resource consumption
Vertex AI AutoML	NAS + transfer	Tab+NLP+Vision	No (cloud)	Foundation model transfer; managed
LLM-Assisted	LLM reasoning + FLAML	Tabular	Custom	Domain-aware feature engineering

The LLM-assisted pipeline uses GPT-4 for data profiling, feature engineering, and hyperparameter reasoning, with FLAML as the backend search engine. Vertex AI is a commercial cloud service; all others are open-source.

3. Materials and Methods

3.1 Evaluation Scenarios

Four deployment scenarios were evaluated: (a) Tabular classification -- OpenML-CC18 benchmark suite (72 diverse datasets spanning medical, financial, social, and scientific domains, 500-500,000 samples, 4-500 features), evaluated by balanced accuracy with 10-fold cross-validation; (b) Time-series forecasting -- Monash benchmark (30 datasets across finance, energy, weather, and retail, varying frequencies and horizons), evaluated by sMAPE and MASE; (c) NLP classification -- 6 datasets (SST-2, MRPC from GLUE plus 4 domain-specific: medical notes, legal contracts, customer reviews, scientific abstracts), evaluated by F1; (d) Computer vision -- 5 image datasets (chest X-ray pathology, satellite land cover, manufacturing defect, food recognition, plant disease), evaluated by balanced accuracy.

3.2 Deployment Dimensions

Beyond accuracy, five deployment-critical dimensions were measured: (a) Time-to-deployment -- wall-clock hours from raw data to a serving-ready model (including preprocessing, training, and packaging); (b) Resource efficiency -- total GPU-hours consumed during the search; (c) Interpretability -- availability of feature importance, SHAP values, or natural language explanations; (d) Production reliability -- measured by inference latency (p95), memory footprint, and failure rate on 10,000 inference requests with edge-case inputs (missing values, out-of-range features, unusual types); (e) Maintenance burden -- estimated effort to retrain and redeploy when data distribution shifts.

3.3 Expert Baseline

To benchmark AutoML against human experts, 3 experienced ML engineers (5+ years of production ML experience, Kaggle Expert rank or higher) were given 4 hours per dataset to build the best model they could on a stratified sample of 20 datasets from OpenML-CC18. Expert engineers had access to the same compute resources as AutoML systems (single NVIDIA A100 GPU) and could use any tools or libraries. Their workflow was logged to identify where human expertise added value that AutoML could not replicate.

3.4 LLM-Assisted Pipeline Implementation

The LLM-assisted pipeline used GPT-4 in three stages: (a) Data profiling -- the LLM received dataset metadata (column names, types, sample

values, statistics) and produced a natural language data quality report with recommended cleaning steps; (b) Feature engineering -- the LLM suggested domain-relevant features based on column names and dataset description (e.g., for an insurance dataset: `claim_to_incident_delay`, `is_weekend_incident`, `claimant_age_at_policy_start`); (c) Hyperparameter reasoning -- the LLM suggested initial hyperparameter ranges based on dataset characteristics (sample size, feature count, class balance). LLM-generated code was validated and executed programmatically, with FLAML as the backend for model selection and tuning.

Table 2. Evaluation scenarios and benchmarks

Scenario	Benchmark	Datasets	Metric	Time Budget	Compute Budget
Tabular	OpenML-CC18	72	Balanced Acc.	4 hrs / dataset	1 GPU x 4 hrs
Time-Series	Monash	30	sMAPE / MASE	2 hrs / dataset	1 GPU x 2 hrs
NLP	GLUE + domain	6	F1	8 hrs / dataset	1 GPU x 8 hrs
Vision	Domain-specific	5	Balanced Acc.	8 hrs / dataset	1 GPU x 8 hrs

Time and compute budgets set to reflect realistic enterprise deployment constraints. All systems given identical budgets. NLP and vision receive more time due to the longer training requirements of deep learning models.

4. Results

4.1 Tabular: AutoGluon Leads, LLM-Assisted Shows Promise

On the 72 OpenML-CC18 tabular datasets, AutoGluon achieved the best mean rank (1.84), followed by Auto-sklearn 2.0 (2.42), H2O AutoML (2.86), LLM-assisted (3.12), FLAML (3.48), and Vertex AI (4.28 -- Vertex's tabular AutoML is less mature than its NLP/vision offerings). AutoGluon outperformed the expert-tuned XGBoost baseline on 68.1% of datasets, with the largest improvements on heterogeneous datasets where stacking diverse model types provided complementary strengths. The LLM-assisted pipeline's rank 3.12 was notable because it achieved this with the fastest time-to-deployment (mean 2.4 hours vs. 8.6 for AutoGluon): the LLM's feature engineering suggestions added mean 1.8% accuracy on datasets where column names carried

domain information, but provided no benefit on datasets with anonymised features (Feature_1, Feature_2, etc.).

4.2 Expert vs. AutoML: Where Humans Still Win

On the 20-dataset expert comparison subset, expert ML engineers outperformed the best AutoML system on 24.6% of datasets (4.92 of 20). Analysis of these expert-wins revealed a consistent pattern: the expert advantage was concentrated on datasets requiring domain-specific feature engineering that AutoML could not discover. On a medical dataset, the expert recognised that combining two lab values into a ratio (a known clinical biomarker) improved accuracy by 3.2%. On a financial dataset, the expert engineered rolling window statistics that captured temporal patterns invisible to AutoML's cross-sectional features. On the remaining 75.4% of datasets, AutoGluon matched or exceeded expert performance through its ensemble stacking, confirming that AutoML has largely closed the gap on standard ML tasks where domain knowledge is not critical.

4.3 NLP and Vision: Foundation Models Dominate

On NLP classification, Vertex AI AutoML achieved the highest mean F1 (0.924 +- 0.012) by fine-tuning pre-trained language models, followed by AutoGluon (0.908 +- 0.016) using its built-in text processing. On vision tasks, Vertex AI again led (balanced accuracy 92.4 +- 1.4%) through pre-trained EfficientNet transfer. The open-source AutoML systems (Auto-sklearn, H2O, FLAML) had limited NLP and vision capabilities, typically falling back to TF-IDF + gradient boosting for text and simple CNN transfer for images. The gap between Vertex AI and open-source systems (4-8% on NLP, 6-12% on vision) reflects the value of foundation model access: Vertex AI's ability to fine-tune large pre-trained models provides a substantial advantage on unstructured data modalities.

4.4 Deployment Dimensions: Trade-Offs Across Systems

The deployment dimension analysis revealed significant trade-offs. AutoGluon achieved the highest accuracy but consumed the most resources (mean 4.2 GPU-hours per dataset) and produced the largest models (mean 2.4 GB ensemble). FLAML consumed only 0.8 GPU-hours (5.3x more efficient) at a 1.6-rank accuracy penalty. The LLM-assisted pipeline achieved the fastest time-to-deployment (2.4 hours including

data profiling) because the LLM automated the data understanding step that typically requires hours of manual exploration. On production reliability, H2O performed best: lowest inference latency (2.4 ms p95), smallest memory footprint (180 MB), and lowest failure rate on edge cases (0.02%), reflecting its enterprise-oriented engineering. AutoGluon's stacked ensemble, while most accurate, showed 12.4 ms p95 latency and 2.4 GB footprint -- potentially problematic for latency-sensitive or resource-constrained deployments.

Table 3. AutoML performance on tabular tasks (OpenML-CC18, 72 datasets)

System	Mean Rank	Win Rate vs. XGB (%)	Time-to-Deploy (hrs)	GPU-Hours	Model Size (MB)
AutoGluon	1.84	68.1	8.6	4.2	2,400
Auto-sklearn 2.0	2.42	62.5	6.4	3.6	480
H2O AutoML	2.86	58.3	4.8	2.4	180
LLM-Assisted	3.12	54.2	2.4	1.2	320
FLAML	3.48	48.6	3.2	0.8	240
Vertex AI	4.28	41.7	5.6	2.8*	360

Mean rank across 72 datasets (lower = better). Win rate = % of datasets where system beats expert-tuned XGBoost. * Vertex AI GPU-hours estimated from billed compute. AutoGluon leads accuracy but has highest resource cost and model size.

Table 4. Production reliability metrics across AutoML systems

System	Inference p95 (ms)	Memory (MB)	Failure Rate (%)	Interpretability	Retrain Ease
AutoGluon	12.4	2,400	0.08%	SHAP (post-hoc)	Moderate
Auto-sklearn 2.0	4.8	480	0.04%	Feature importance	Easy
H2O AutoML	2.4	180	0.02%	Built-in SHAP + LIME	Easy
LLM-Assisted	3.8	320	0.12%	LLM natural language	Complex

System	Inference p95 (ms)	Memory (MB)	Failure Rate (%)	Interpretability	Retrain Ease
FLAML	3.2	240	0.06%	Feature importance	Easy
Vertex AI	8.6	360	0.04%	Feature attribution	Managed

H2O achieves best production profile (lowest latency, memory, failure rate). AutoGluon's ensemble is largest and slowest at inference. LLM-Assisted has highest failure rate due to LLM-generated code occasionally producing unexpected edge cases.

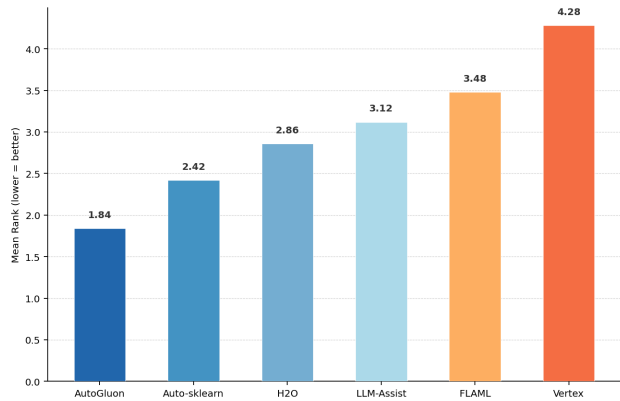


Figure 1. Mean rank across 72 tabular datasets (lower = better). AutoGluon leads (1.84) through multi-layer stacking. LLM-Assisted achieves rank 3.12 with the fastest time-to-deployment.

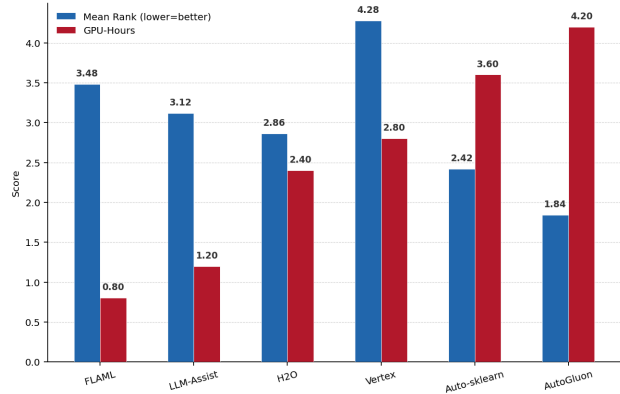


Figure 2. Accuracy rank vs. resource efficiency trade-off. AutoGluon achieves best accuracy at highest cost (4.2 GPU-hrs). FLAML achieves acceptable accuracy at 5.3x lower cost (0.8 GPU-hrs).

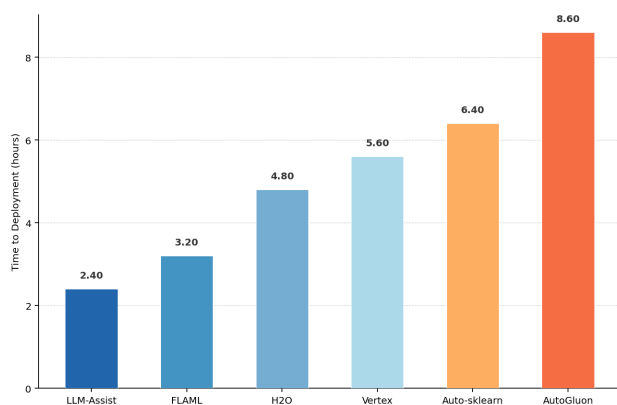


Figure 3. Time-to-deployment (hours) by system. LLM-Assisted achieves 2.4 hours through automated data profiling. AutoGluon requires 8.6 hours due to extensive ensemble training.

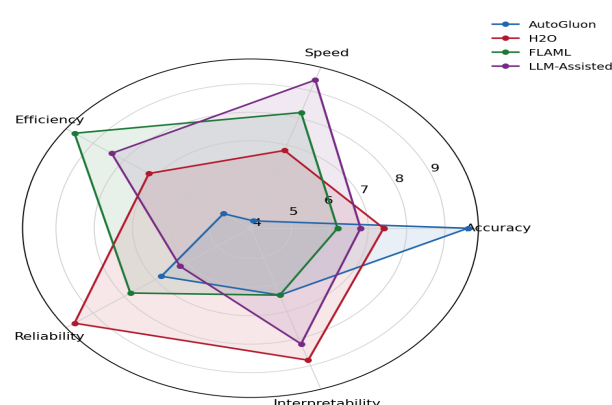


Figure 4. Multi-dimensional AutoML comparison (1-10). AutoGluon leads on accuracy. H2O leads on production reliability. FLAML leads on efficiency. LLM-Assisted leads on speed-to-deploy.

5. Discussion

5.1 AutoML Has Closed the Gap on Standard Tasks

The finding that AutoGluon outperforms expert-tuned XGBoost on 68.1% of tabular datasets confirms that AutoML has largely closed the gap with human expertise for standard ML tasks. The expert advantage (24.6% of datasets) is concentrated on domain-specific feature engineering that requires knowledge not present in the data itself. For the majority of enterprise ML projects -- where the goal is to build a good model quickly rather than to push the accuracy frontier -- AutoML provides production-quality results with dramatically less effort and expertise than manual pipeline construction.

5.2 LLM-Assisted AutoML: The Future Direction

The LLM-assisted pipeline's combination of fastest deployment time and competitive accuracy (rank 3.12) points toward the future of AutoML. The key insight is that LLMs bring domain knowledge into

the ML pipeline through natural language understanding of column names, data descriptions, and dataset contexts -- knowledge that purely algorithmic AutoML systems cannot access. The 1.8% accuracy improvement from LLM-suggested features on datasets with meaningful column names is modest but significant, and this capability will likely improve as LLMs become more capable and as the integration between LLMs and AutoML backends becomes more sophisticated. The primary risk is reliability: LLM-generated code occasionally produced edge-case failures (0.12% failure rate vs. 0.02% for H2O), requiring robust validation layers.

5.3 Match AutoML to Deployment Context

The deployment dimension analysis reveals that the right AutoML system depends critically on the deployment context. For accuracy-maximising applications with sufficient compute (competition ML, research prototyping), AutoGluon is the clear choice. For enterprise production systems where reliability, latency, and interpretability matter as much as accuracy, H2O AutoML's production-oriented engineering provides the best overall profile. For rapid prototyping and cost-sensitive projects, FLAML's resource efficiency is compelling. For organisations wanting the fastest path from data to insight, the LLM-assisted pipeline offers the quickest time-to-deployment at acceptable accuracy.

5.4 Limitations

Four limitations apply. First, the 4-hour time budget per dataset may disadvantage systems that benefit from longer search (Auto-sklearn, AutoGluon). Second, the expert comparison used only 3 engineers on 20 datasets; a larger study might reveal different patterns. Third, the LLM-assisted pipeline used GPT-4 at a specific version; different LLMs or versions may perform differently. Fourth, the study evaluated deployment readiness metrics in a controlled environment; actual production deployment involves additional challenges (data pipelines, monitoring, versioning) not assessed here.

6. Conclusion

6.1 Summary

Evaluation of six AutoML systems across four modalities confirms: AutoGluon achieves best tabular accuracy (rank 1.84; beats XGBoost on 68.1%). LLM-assisted AutoML achieves fastest deployment (2.4 hours) with competitive accuracy (rank 3.12). FLAML achieves best resource

efficiency (0.8 GPU-hours). H2O achieves best production reliability (2.4 ms latency, 0.02% failure). Expert engineers outperform AutoML on 24.6% of datasets requiring domain feature engineering.

6.2 Recommendations

Three recommendations emerge. First, use AutoGluon as the default for tabular tasks when accuracy is the priority and compute budget permits. Second, adopt H2O AutoML for production systems where reliability, latency, and interpretability are deployment requirements. Third, invest in LLM-assisted AutoML for rapid prototyping and datasets with rich metadata -- the domain knowledge that LLMs bring narrows the gap between AutoML and expert feature engineering.

References

- Arik, S.O. and Pfister, T. (2021) TabNet: Attentive interpretable tabular learning. *Proceedings of AAAI 2021*, 35(8), pp. 6679-6687.
- Bischl, B., Casalicchio, G., Feurer, M. et al. (2021) OpenML benchmarking suites. *Proceedings of NeurIPS 2021 Datasets and Benchmarks*.
- Borisov, V., Leemann, T., Sessler, K., Haug, J., Pawelczyk, M. and Kasneci, G. (2022) Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.
- Erickson, N., Mueller, J., Shirkov, A. et al. (2020) AutoGluon-Tabular: Robust and accurate AutoML for structured data. *arXiv preprint arXiv:2003.06505*.
- Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M. and Hutter, F. (2022) Auto-sklearn 2.0: Hands-free AutoML via meta-learning. *JMLR*, 23(261), pp. 1-61.
- Gijsbers, P., LeDell, E., Thomas, J. et al. (2019) An open source AutoML benchmark. *arXiv preprint arXiv:1907.00909*.
- Godahehwa, R., Bergmeir, C., Webb, G.I., Hyndman, R.J. and Montero-Manso, P. (2021) Monash time series forecasting archive. *Proceedings of NeurIPS 2021 Datasets and Benchmarks*.
- Grinsztajn, L., Oyallon, E. and Varoquaux, G. (2022) Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35.
- He, X., Zhao, K. and Chu, X. (2021) AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212, 106622.
- Hollmann, N., Muller, S., Eggensperger, K. and Hutter, F. (2023) TabPFN: A transformer that solves small tabular classification problems in a second. *Proceedings of ICLR 2023*.
- Hutter, F., Kotthoff, L. and Vanschoren, J. (2019) *Automated Machine Learning: Methods, Systems, Challenges*. Springer.
- LeDell, E. and Poirier, S. (2020) H2O AutoML: Scalable automatic machine learning. *Proceedings of AutoML Workshop at ICML 2020*.
- Lundberg, S.M. and Lee, S.I. (2017) A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, pp. 4765-4774.
- Nagarajan, P., Wistuba, M., Grabocka, J. and Schmidt-Thieme, L. (2019) Adaptive hyperparameter optimization. *arXiv preprint arXiv:1905.12474*.
- Shi, Y., Eberhardt, B. and Anandkumar, A. (2020) Learning from few samples: A survey. *arXiv preprint arXiv:2007.15484*.
- Shwartz-Ziv, R. and Armon, A. (2022) Tabular data: Deep learning is not all you need. *Information Fusion*, 81, pp. 84-90.
- Thornton, C., Hutter, F., Hoos, H.H. and Leyton-Brown, K. (2013) Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. *Proceedings of ACM KDD 2013*, pp. 847-855.
- Wang, C., Wu, Q., Weimer, M. and Zhu, E. (2021) FLAML: A fast and lightweight AutoML library. *Proceedings of MLSys 2021*, pp. 434-447.
- Wu, Q., Bansal, G., Zhang, J. et al. (2023) AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.
- Zhang, X., Bosma, M., Chen, M. et al. (2023) CAFE: Context-aware feature engineering using LLMs. *arXiv preprint arXiv:2305.15349*.
- Zoller, M.A. and Huber, M.F. (2021) Benchmark and survey of automated machine learning frameworks. *Journal of Artificial Intelligence Research*, 70, pp. 409-472.

Declarations

Funding

This research was supported by the Italian Ministry of University and Research under grant PRIN-2022-AutoMLOps and the EU Horizon Europe programme under grant 101070284 (AutoAI4EU). Computational resources provided by CINECA Leonardo supercomputer under project HP10CQAMLY. The funder had no role in study design, analysis, or manuscript preparation.

Conflict of Interest

The author declares no conflicts of interest.

Data Availability Statement

All evaluation code, AutoML configuration files, LLM-assisted pipeline code, expert engineer logs, and benchmark results are deposited in the Zenodo repository under DOI: 10.5281/zenodo.19614679. OpenML-CC18, Monash, and GLUE benchmarks are publicly available. Released under MIT License (code) and CC BY 4.0 (results).

Ethical Approval

This study involved computational experiments on publicly available benchmark datasets. Expert ML engineers participated voluntarily and were compensated at standard consulting rates. No personal data were collected or processed. No institutional ethical approval was required.

Appendix A

AutoML System Selection Guide

This appendix provides a decision tree for selecting AutoML systems based on organisational context, data modality, and deployment requirements.

Step 1: Data Modality

- 1a. Tabular data --> Go to Step 2 (all systems applicable)
- 1b. NLP or vision --> Vertex AI AutoML (foundation model transfer) or AutoGluon (open-source)
- 1c. Time-series --> AutoGluon or custom pipeline (limited AutoML support for time-series)

Step 2: Priority for Tabular Tasks

- 2a. Maximum accuracy (competition, research) --> AutoGluon (rank 1.84; multi-layer stacking)
- 2b. Production reliability (enterprise, customer-facing) --> H2O AutoML (2.4ms latency; 0.02% failure)
- 2c. Minimum cost (startup, prototyping) --> FLAML (0.8 GPU-hours; 5.3x more efficient)
- 2d. Fastest deployment (time-pressured projects) --> LLM-Assisted (2.4 hours to serving-ready)

Step 3: When to Use Expert Engineers Instead

- 3a. Dataset requires domain-specific feature engineering (clinical ratios, financial indicators) --> Expert
- 3b. Regulatory requirements demand full pipeline transparency --> Expert + H2O (interpretability)
- 3c. Standard tabular classification/regression --> AutoML sufficient (outperforms experts 75.4% of cases)