CS4824: Capstone Project Proposal

Matthew Nissen

due 09/19/2025

Problem Definition

Machine learning has become central to modern applications, yet building effective models remains a complex, resource-intensive task requiring expertise in preprocessing, feature engineering, model selection, and hyperparameter tuning. This creates a significant barrier for practitioners who need accurate predictions but lack the specialized training to design optimized pipelines. Automated Machine Learning (AutoML) aims to lower this barrier by systematizing and automating the end-to-end process of model development.

While industry systems such as Google AutoML and open-source frameworks like Auto-sklearn have advanced the field, most available solutions either require large-scale compute resources or operate as "black boxes" with limited transparency. There is a clear need for educationally motivated, resource-conscious AutoML systems that both demonstrate the core mechanisms of automation and explore extensions toward more sophisticated functionality (e.g., Bayesian optimization, meta-learning, ensemble construction).

Research Questions:

- Framework Design: How can a lightweight AutoML framework be built to automate the essential components of model development - data preprocessing, model selection, and hyperparameter tuning - while remaining transparent and computationally efficient?
- Framework Evaluation: How well does the baseline AutoML system perform across diverse datasets, and what strengths and limitations emerge when compared to traditional manually tuned pipelines?
- Advanced Extensions: To what extent can advanced AutoML techniques (e.g., Bayesian optimization, meta-learning, ensemble construction) be approximated or adapted within the constraints of a semester project to emulate the capabilities of systems like Google AutoML?

This project is significant because it not only reinforces core concepts from the course (evaluation, generalization, optimization, algorithmic design) but also pushes beyond the standard curriculum into state-of-the-art approaches in AutoML. The novelty lies in constructing a system that bridges pedagogical clarity and cutting-edge ambition: a framework that is both educational and exploratory, aiming to approximate Google AutoML-level sophistication within an academic, resource-limited environment.

Literature Review

Automated Machine Learning (AutoML) has developed into a diverse field of methods for reducing the manual effort of building machine learning pipelines. Early systems such as Auto-WEKA established the foundation by framing the challenge as the combined algorithm selection and hyperparameter optimization (CASH) problem, solved with Bayesian optimization (Thornton et al., 2013). This approach highlighted the potential of systematically searching across models and settings rather than relying on ad hoc tuning. Building on that idea, auto-sklearn incorporated meta-learning to warm-start the search process and introduced automatic ensembling to improve robustness, setting a strong benchmark for general-purpose AutoML on tabular data (Feurer et al., 2015).

Other projects explored alternative strategies. TPOT applied genetic programming to evolve pipelines of preprocessing steps and estimators, demonstrating the flexibility of evolutionary search while also exposing the trade-offs in computation time (Olson et al., 2016). Auto-Keras extended the scope further by targeting deep learning with neural architecture search, providing greater modeling power but at the cost of heavy resource requirements (Jin et al., 2019). More recent surveys and overviews (He et al., 2021; van der Putten & van Someren, 2019) emphasize the progress across these approaches while also underscoring open challenges, particularly around efficiency, interpretability, and the balance between broad applicability and practical constraints.

Despite these advances, current frameworks tend to assume access to substantial compute resources, primarily benchmark against each other, and often integrate many advanced techniques at once. This makes it difficult to isolate the impact of individual design choices. The project proposed here addresses this gap by: (1) implementing a lightweight AutoML framework designed to operate within semester-level compute limits, (2) explicitly benchmarking against manually tuned baselines drawn from coursework, and (3) testing the incremental value of adding a single advanced extension, such as Bayesian optimization.

View References section for cited papers.

Methodology

The proposed framework will be developed in two stages: a baseline AutoML system that automates the core steps of pipeline construction, and an extension stage that incorporates one advanced search technique inspired by large-scale AutoML systems.

Baseline System.

The foundation of the framework will focus on automating three essential tasks: data preprocessing, model selection, and hyperparameter tuning. Preprocessing will include common transformations such as handling missing values, encoding categorical features, and scaling numerical features, implemented in a modular way so that each step can be swapped in or out. A curated set of model families will be supported, including linear models (logistic regression, ridge regression), tree-based methods (decision trees, random forests, gradient boosting), and a simple feed-forward neural network. Hyperparameter search at the baseline level will begin with grid and random search, both of which are computationally feasible under academic constraints and offer interpretable search behavior.

Advanced Extension.

To push beyond a purely baseline framework, the project will explore the integration of an advanced search method such as Bayesian optimization. Unlike grid or random search, Bayesian optimization uses a probabilistic surrogate model to guide exploration toward promising regions of the hyperparameter space, often yielding better results under limited budgets. This extension is deliberately scoped as incremental: the aim is not to replicate the full sophistication of Google AutoML, but to approximate one of its capabilities in a resource-conscious way, and to measure the added value relative to the baseline.

Experimental Design.

The framework will be evaluated on several publicly available tabular datasets (e.g., UCI repository, OpenML). Each dataset will be split into training, validation, and test sets to ensure proper evaluation. The AutoML system's performance will be compared against:

- Manual baselines: pipelines constructed with standard algorithms and tuned using simple heuristics from coursework.
- Baseline AutoML framework: automated preprocessing, model selection, and random/grid search.
- Extended AutoML framework: baseline system plus Bayesian optimization (or another advanced extension if time allows).

Performance will be measured with metrics appropriate to the task (e.g., accuracy or F1 for classification, RMSE for regression) along with resource-focused metrics such as runtime and number of model evaluations. This dual focus ensures the evaluation highlights both predictive power and efficiency, which are central to the project's goals.

Justification.

This design ensures feasibility within the semester: grid/random search and a modest set of models guarantee a working baseline, while Bayesian optimization offers an achievable extension. By structuring the evaluation to include both manual baselines and incremental improvements, the project directly addresses the identified gaps in prior work, namely the lack of resource-conscious, baseline-centered benchmarking and the difficulty of isolating the impact of individual extensions.

Data & Resources

Datasets.

The system will be evaluated on multiple publicly available tabular datasets from the UCI Machine Learning Repository and OpenML, covering both classification and regression. Examples include Iris (sanity checks), Adult Income (categorical/numerical mix), Wine Quality (regression), and larger OpenML datasets such as Credit Default or Australian dataset. These datasets are widely used in benchmarking AutoML systems, ensuring results are comparable to prior work.

Role of Data Analysis.

In a traditional ML project, preliminary exploration (e.g., checking distributions, missing values, class balance) is done manually before modeling. In this project, such analysis is itself one of the goals of the AutoML system. Rather than cleaning datasets by hand, the framework will implement preprocessing modules that automatically detect and handle common challenges (missing data, categorical encoding, feature scaling, class imbalance). This design choice reflects the project's emphasis on building the process rather than manually engineering solutions for each dataset.

Compute Resources.

The framework will be developed to run under academic compute limits, primarily on laptops or cloud notebook environments (e.g., Jupyter Notebook/Google Colab). The baseline search strategies (grid and random) are lightweight and interpretable, while the advanced extension (Bayesian optimization) can be implemented efficiently using existing libraries such as Optuna. Search budgets (e.g., number of model trials per dataset) will be capped to keep experiments tractable.

Preparation.

To support development, datasets will initially be loaded in raw form and inspected at a high level (row/column counts, feature types) to guide the design of preprocessing modules. However, full exploratory analysis and model fitting will be handled by the AutoML framework itself once implemented. This shifts the focus from solving individual datasets to designing a generalizable, modular process that can handle a variety of tasks.

Timeline & Feasibility

Week 2 (Sept 22–28):Preprocessing Module Implementation

- Develop automated detection and handling of missing values, categorical encoding, and feature scaling.
- Validate preprocessing independently on small datasets (Iris, Wine Quality).
- Milestone: Preprocessing modules functional.

Week 3 (Sept 29–Oct 5): Model Wrappers

- Implement wrappers for logistic regression, decision trees, random forests, gradient boosting, and a simple neural net.
- Ensure standardized interfaces for training, prediction, and evaluation.
- Milestone: Model zoo established.

Week 4 (Oct 6–12): Baseline Search Strategies & Pipeline Integration

- Implement grid search and random search.
- Integrate preprocessing + models + search into a single end-to-end AutoML pipeline.
- Run first full-system tests on 1–2 datasets.
- Milestone: Baseline AutoML operational.

Week 5 (Oct 13–19): Evaluation Framework

- Add consistent metrics (accuracy, F1, RMSE, runtime).
- Log search paths, intermediate results, and final selections.
- Benchmark baseline AutoML vs manually tuned pipelines.
- Milestone: Baseline AutoML evaluated against manual baselines.

Week 6 (Oct 20–26): Advanced Extension — Bayesian Optimization

- Integrate Bayesian optimization (Optuna or scikit-optimize).
- Compare performance/runtime vs random/grid search on initial datasets.
- Milestone: Extended AutoML functional with Bayesian optimization.

Week 7 (Oct 27–Nov 2): Systematic Evaluation

- Run comparative experiments across all selected datasets.
- Collect baseline vs extended results; analyze strengths and weaknesses.
- Milestone: Dataset-by-dataset evaluation completed.

Week 8 (Nov 3–9): Stretch Goal #1 — Ensemble Construction

- Implement simple model stacking or blending.
- Evaluate incremental performance improvements relative to Bayesian optimization.
- Milestone: Ensemble strategies tested and benchmarked.

Week 9 (Nov 10–16): Stretch Goal #2 — Meta-Learning Warm Starts

- Implement a metadata-driven strategy: use dataset characteristics (e.g., # features, class balance) to suggest starting hyperparameters/models.
- Compare performance vs non–warm-start runs.
- Milestone: Meta-learning prototype operational.

Week 10 (Nov 17–23): Stretch Goal #3 — Neural Architecture Search / Genetic Programming (Exploratory)

- Experiment with a simplified NAS (Auto-Keras style) for small neural networks, or genetic programming for pipeline evolution (TPOT-inspired).
- Focus on feasibility within compute limits.
- Milestone: Proof-of-concept results for one advanced technique.

Week 11 (Nov 24–30): Consolidation & Robustness Testing

- Re-run experiments across multiple datasets for consistency.
- If stretch goals underperform, pivot to refining existing modules.
- Milestone: Stable results across baseline + extensions.

Week 12 (Dec 1–10): Final Deliverables

- Draft and polish final report with figures, search traces, and performance tables.
- Prepare spotlight presentation slides and demo.
- Milestone: Complete capstone submission and presentation.

References

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. In Advances in Neural Information Processing Systems (NeurIPS). Retrieved from https://www.automl.org/wp-content/uploads/2019/05/AutoML_Book_Chapter6.pdf

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning (auto-sklearn preprint). Retrieved from https://ml.informatik.uni-freiburg.de/wp-content/uploads/papers/15-NIPS-auto-sklearn-preprint.pdf

Olson, R. S., Bartley, N., Urbanowicz, R. J., & Moore, J. H. (2016). Evaluation of a tree-based pipeline optimization tool for automating data science. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) (pp. 485–492). ACM. https://proceedings.mlr.press/v64/olson_tpot_2016.html

Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013). Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 847–855). ACM. http://www.cmap.polytechnique.fr/~nikolaus.hansen/proceedings/2016/GECCO/proceedings/p485.pdf

Jin, H., Song, Q., & Hu, X. (2019). Auto-Keras: An efficient neural architecture search system. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 1946–1956). https://dl.acm.org/doi/pdf/10.5555/3586589.3586850

He, X., Zhao, K., & Chu, X. (2021). AutoML: A survey of the state-of-the-art. Knowledge-Based Systems, 212, 106622. https://www.sciencedirect.com/science/article/abs/pii/S0950705120307516

van der Putten, P., & van Someren, M. (2019). Chapter 6: Automatic machine learning. In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), Automated Machine Learning: Methods, Systems, Challenges. Springer. https://www.automl.org/wp-content/uploads/2019/05/AutoML_Book_Chapter6.pdf