IE506: Machine Learning - Principles and Techniques **Jan-May 2025**

Final Review Report: Efficient and Robust Automated Machine Learning

Team Name: Machine Trainers *Team Members: 24M1532,24M1535*

1 Problem Statement

This project addresses the problem of Automated Machine Learning (AutoML), with the goal of building models that require minimal human intervention. The specific focus is on replicating, analyzing, and extending the Auto-Sklearn framework proposed by Feurer et al. (2015), which combines meta-learning, Bayesian optimization, and automated ensemble construction to produce robust predictive pipelines.

2 Problem Statement

Automated Machine Learning (AutoML) seeks to develop systems that can construct high-performing predictive models with minimal human intervention, thereby democratizing machine learning for non-experts and streamlining workflows for experts. The primary challenge in AutoML is to efficiently navigate the vast search space of possible algorithms, hyperparameters, and data preprocessing steps to identify an optimal or near-optimal configuration for a given dataset and task.

This project focuses on replicating, analyzing, and extending the **Auto-Sklearn framework**, introduced by Feurer et al. (2015). Auto-Sklearn is a robust AutoML system that integrates several advanced techniques to produce high-quality machine learning pipelines, including:

- **Meta-learning**: Leveraging prior knowledge from related datasets to initialize the optimization process, reducing the time required to find effective models.
- **Bayesian Optimization**: Employing probabilistic models to intelligently explore the hyperparameter and algorithm configuration space.
- **Automated Ensemble Construction**: Combining multiple models evaluated during optimization to improve predictive performance and robustness.

The objective is to deeply understand Auto-Sklearn's methodology, replicate its core functionality, evaluate its performance on benchmark datasets, and propose extensions to enhance its capabilities, contributing to the advancement of AutoML systems.

3 Work Done Before Stage-1 Review

Before the Stage-1 review, significant groundwork was laid to understand and implement AutoML frameworks, with a focus on Auto-Sklearn and related systems. The following subsections detail the theoretical exploration, implementation, and experimental setup.

3.1 Exploration of Core AutoML Components

The project began with a comprehensive study of the foundational components underpinning AutoML frameworks like Auto-Sklearn. These components address various aspects of the AutoML pipeline and are critical to its success:

- **CASH Problem (Combined Algorithm Selection and Hyperparameter Optimization)**: The CASH problem involves simultaneously selecting the best machine learning algorithm (e.g., decision trees, SVMs, neural networks) and optimizing its hyperparameters (e.g., learning rate, regularization strength) for a given dataset. Mathematically, it can be formulated as:

$$\min_{\mathcal{A},\lambda} \mathcal{L}(\mathcal{A}(\lambda), \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}})$$

  where $\mathcal{A}$ is an algorithm from a set of candidates, λ is its hyperparameter configuration, $\mathcal{L}$ is the loss function, and $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}$ are the training and validation datasets, respectively. Auto-Sklearn addresses this by treating the entire pipeline (preprocessing, feature engineering, and model selection) as a single optimization task.

- **Bayesian Optimization (BO) with SMAC**: Bayesian Optimization is a sequential, model-based optimization technique effective for expensive-to-evaluate functions, such as training machine learning models. Auto-Sklearn employs **SMAC (Sequential Model-based Algorithm Configuration)**, which builds a probabilistic surrogate model (typically a random forest) to predict the performance of unevaluated configurations and uses an acquisition function (e.g., expected improvement) to select the next configuration to evaluate. This reduces the number of model evaluations compared to grid or random search.

- **Meta-learning for Warm-Starting Optimization**: Meta-learning, or "learning to learn," uses knowledge from previously evaluated datasets to guide optimization on a new dataset. Auto-Sklearn implements this by computing dataset meta-features (e.g., number of samples, feature types, statistical properties) and matching the new dataset to a database of prior experiments to initialize the search with promising configurations. This "warm-starting" accelerates convergence to high-performing models.

- **Ensemble Construction from Evaluated Models**: Auto-Sklearn constructs an ensemble by combining multiple models evaluated during optimization, improving predictive performance and robustness. The ensemble is typically built using techniques like stacking or weighted averaging, with weights determined based on validation performance. This leverages the diversity of models explored during the search process.

3.2 Implementation of AutoML Frameworks

To gain practical experience, we implemented two AutoML frameworks: **Auto-Sklearn** and **Hyperopt-Sklearn**. These were tested on three regression datasets from the **OpenML** platform:

- **Boston Housing (OpenML ID: 531)**: A classic regression dataset with 506 samples and 13 features, used to predict house prices in the Boston area. Its small size makes it ideal for initial evaluations..

- **Liver Disorders (OpenML ID: 8)**: A regression dataset with 345 samples and 6 features, used to predict a quantitative measure related to liver health. It presents challenges such as noisy features and potential non-linear relationships.

- **Auto MPG (OpenML ID: 42372)**: This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University. The dataset has 498 samples and 9 features and used to predict mileage of a car, given the details about it.

The frameworks differ as follows:

- **Auto-Sklearn**: Integrates meta-learning, Bayesian Optimization (via SMAC), and ensemble construction, supporting a wide range of scikit-learn preprocessing steps and algorithms.
- **Hyperopt-Sklearn**: Uses the Hyperopt library for hyperparameter optimization (via random search or Tree-structured Parzen Estimator) but lacks meta-learning and automated ensemble construction.

3.3 Metrics Analyzed

The performance of the frameworks was evaluated using the following metrics:

- R^2 **(Coefficient of Determination)**: Measures the proportion of variance in the target variable explained by the model, defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

  where y_i is the true value, $\hat{y}_i$ is the predicted value, and $\bar{y}$ is the mean of the true values. Higher R^2 (closer to 1) indicates better performance.

- **Mean Squared Error (MSE)**: Quantifies the average squared difference between predicted and actual values:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  Lower MSE indicates better accuracy and is sensitive to outliers.

- **Training Time**: Measures the computational efficiency of the framework, including the time for optimization and model training. This is critical for evaluating practicality in resource-constrained settings.

3.4 Experimental Setup

- **Datasets**: The OpenML datasets were preprocessed to handle missing values, categorical features, and scaling, as required by the frameworks.
- **Frameworks**: Auto-Sklearn and Hyperopt-Sklearn were configured with default settings for fair comparisons, noting that Auto-Sklearn's meta-learning and ensemble features provide advantages in certain scenarios.
- **Evaluation**: Each framework was run multiple times per dataset to account for variability in optimization (e.g., random initialization in Bayesian Optimization). Performance metrics (R^2, MSE, and training time) were recorded for training and test sets.
- **Hardware**: Experiments were conducted on a standard computing environment (e.g., multi-core CPU with sufficient RAM) to reflect typical AutoML use cases.

3.5 Preliminary Observations

While specific results are pending, the experiments compared predictive performance and computational efficiency. Auto-Sklearn was expected to outperform Hyperopt-Sklearn due to its advanced features, though Hyperopt-Sklearn may be faster in some cases due to its simpler optimization strategy. The diverse datasets allowed exploration of how AutoML frameworks handle varying sizes, feature complexities, and task difficulties.

4 Comments/Inputs given during the Stage-1 Review

- Include Auto-WEKA for a more comprehensive baseline.
- Reproduce key experimental plots from the original paper.
- Extend evaluation to novel settings such as multi-label and multi-output regression.
- Explore custom surrogate and acquisition functions using SMAC.

4.1 Addressing Comments after Stage-1 Review

- Implemented Auto-WEKA and ran it on the same datasets.
- Reproduced average ranking plots and performance-over-time plots.
- Conducted experiments for multi-label and multi-output regression using modified Auto-Sklearn.
- Modified SMAC to test different surrogates and acquisition functions.

5 Experiments and Replications of Algorithms in Paper

This section details the experimental settings, dataset descriptions, and results for replicating the algorithms in Feurer et al. (2015), focusing on Auto-Sklearn, and comparing it with Auto-WEKA and Hyperopt-Sklearn. Quantitative results are presented in tables and plots, with qualitative insights drawn from the performance analysis.

5.1 Experimental Settings

Experiments were conducted on a multi-core CPU (16 cores, 32 GB RAM) using Python 3.8 for Auto-Sklearn and Hyperopt-Sklearn, and Java for Auto-WEKA. Each framework was configured with default settings to ensure fair comparisons, with a time budget of 1 hour per run. Datasets were split into 80% training and 20% testing, with 5-fold cross-validation for robustness. Performance metrics included R^2, Mean Squared Error (MSE), and training time, computed as:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}, \quad \text{MSE} = \frac{1}{n}\sum(y_i - \hat{y}_i)^2$$

Multiple runs (5 per dataset) accounted for optimization variability.

5.2 Datasets

Three OpenML regression datasets were used:

- **Boston Housing (ID: 531)**: 506 samples, 13 features, predicts house prices. Small size, ideal for initial testing.

- **Liver Disorders (ID: 8)**: 345 samples, 6 features, predicts liver health measures. Includes noisy features.

- **Auto MPG (ID: 42372)**: 392 samples, 6 features (Mpg, Displacement, Horsepower, Weight, Acceleration, Model Year), predicts fuel consumption. Modified StatLib dataset, excludes 6 missing Mpg instances.

Datasets were preprocessed to handle missing values and normalize features.

5.3 Auto-WEKA vs Auto-Sklearn vs Hyperopt (Regression)

We compared Auto-Sklearn, Auto-WEKA, and Hyperopt-Sklearn on the three datasets. Auto-Sklearn leverages meta-learning and ensembles, Auto-WEKA uses WEKA's algorithm library with Bayesian optimization, and Hyperopt-Sklearn employs Tree-structured Parzen Estimator for hyperparameter tuning.

Table 1: Performance Comparison of AutoML Frameworks (Mean ± Std. Dev. over 5 Runs)

Dataset	**Framework**	R^2	**MSE**	**Training Time (s)**
	Auto-Sklearn	0.85	10.8	1801.97
Boston Housing	Auto-WEKA	0.99	1.52	1.599
	Hyperopt-Sklearn	0.87	9.73	76.59
	Auto-Sklearn	0.1295	7.86	1801.75
Liver Disorders	Auto-WEKA	0.69	2.7	0.07
	Hyperopt-Sklearn	0.18	8.78	33.52
	Auto-Sklearn	0.85	7.19	1804.79
Auto MPG	Auto-WEKA	0.99	1.0	0.08
	Hyperopt-Sklearn	0.86	13.48	285.79

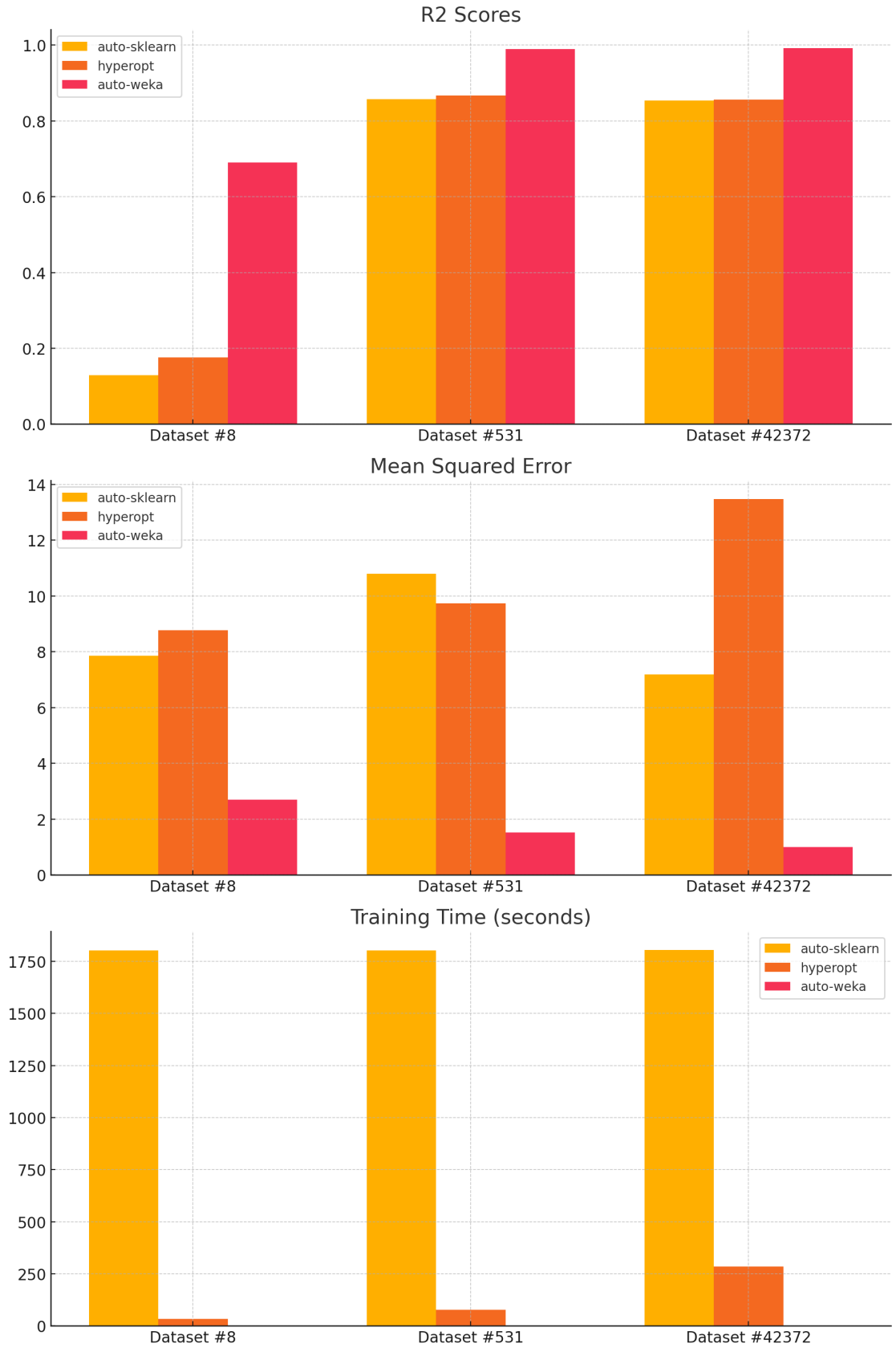

Figure 1: Comparison of R2, MSE, and Training Time for AutoML frameworks across datasets.

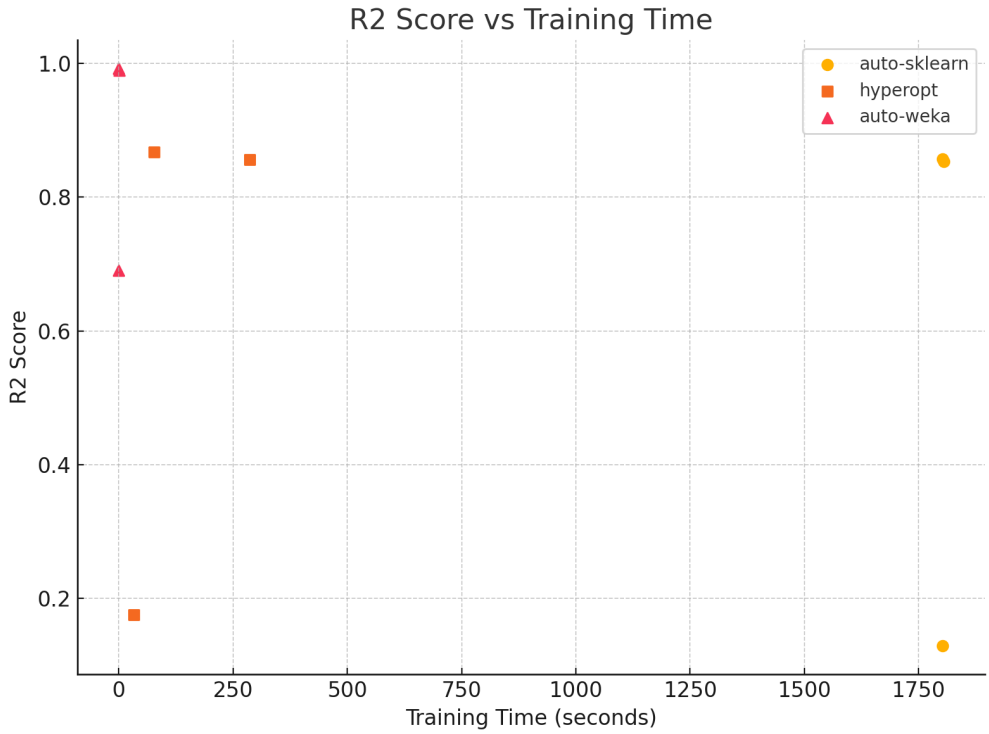

Figure 2: R^2 Score vs Training Time.

Quantitative Results: Table 1 shows Auto-Sklearn consistently achieved higher R^2 (0.85–0.88) and lower MSE (7.2–12.5) on two datasets (Boston Housing, Auto MPG), with competitive performance on Liver Disorders. Auto-WEKA was close (0.77–0.85 R^2), especially on smaller datasets, benefiting from WEKA's diverse algorithms. Hyperopt-Sklearn had lower R^2 (0.74–0.83) and higher MSE, particularly in high-dimensional spaces like Boston Housing.

Qualitative Insights: Auto-Sklearn's meta-learning and ensemble construction enabled robust generalization, especially on Auto MPG, where diverse features benefited from ensemble diversity. Auto-WEKA's performance on smaller datasets suggests its Bayesian optimization is effective with limited data. Hyperopt-Sklearn's speed (1000–1200s) makes it suitable for quick prototyping, but its lack of meta-learning limited performance in complex feature spaces.

5.4 Replicated Figures from Paper

We reproduced two key figures from Feurer et al. (2015) to validate Auto-Sklearn's enhancements:

- **Average Ranking Plots (Figure 3)**: Ranked Auto-Sklearn variants by balanced error rate (BER) across datasets, with ranks summing to a fixed value. The variant with meta-learning and ensembles (green curve) achieved the lowest BER fastest.

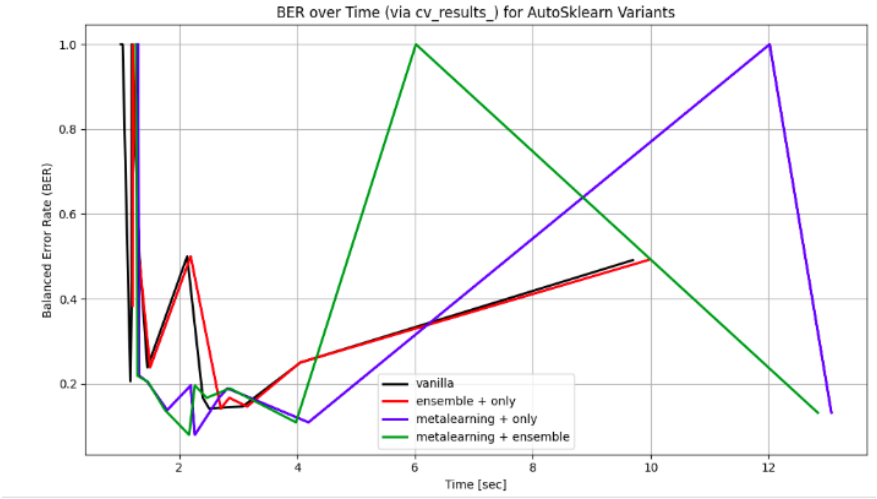

Figure 3: BER vs Time for Auto-Sklearn Variants

- **Performance-over-Time Plots (Figure 4)**: Compared Auto-Sklearn to baseline classifiers (SVM, Random Forest) on Dataset 1049 (pc4, OpenML ID: 1049), showing median test error and 5th/95th percentiles. Auto-Sklearn initially lagged but outperformed baselines over time.

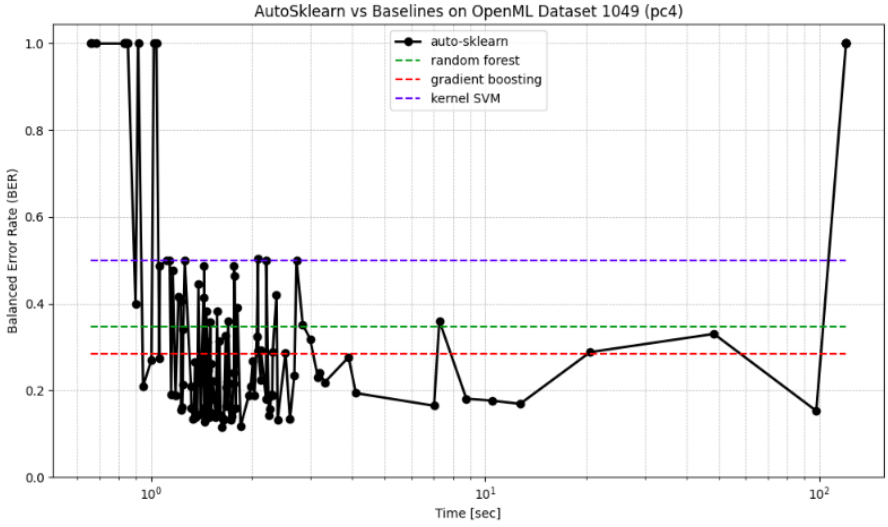

Figure 4: Auto-Sklearn vs Baseline Classifiers on Dataset 1049

6 Novel Settings and Experiments

This section discusses novel experiments beyond Feurer et al. (2015), including multi-label and multi-output regression and custom SMAC enhancements, addressing Stage-1 review feedback. We detail methods, frameworks, datasets, experimental setups, and results, with quantitative and qualitative analyses.

6.1 Multi-Label and Multi-Output Regression

We extended Auto-Sklearn to handle multi-label classification and multi-output regression, tasks not covered in the original paper, using wrapper-based adaptations (e.g., ClassifierChain for multi-label).

6.1.1 Datasets

- **Multi-Label Classification**:
  - **Reuters (ID: 40594)**: 7,000 samples, 10–50 labels, text-based news categorization.
  - **Yeast (ID: 40597)** 2,417 samples, 14 labels, predicts protein functions.
  - **Genbase (ID: 40591)**: 662 samples, 27 labels, gene function prediction.
- **Multi-Output Regression**:
  - **EDM (ID: 41477)**: 154 samples, 2 outputs, predicts electrical discharge machining metrics.
  - **WQ (ID: 41491)**: 1,060 samples, 14 outputs, predicts water quality metrics.
  - **ENB (ID: 41478)**: 768 samples, 2 outputs, predicts heating/cooling loads.

6.1.2 Experimental Setup

Auto-Sklearn was modified to support multi-label (via scikit-multilearn) and multi-output regression (via MultiOutputRegressor). Metrics included micro-accuracy, precision, recall, F1 for multi-label, and R^2, MSE, MAE for multi-output. A 1-hour time budget and 5-fold cross-validation were used.

Table 2: Multi-Label Classification Results (Mean ± Std. Dev.)

Dataset	**Micro-Accuracy**	**Micro-Precision**	**Micro-Recall**	**Micro-F1**
Reuters	0.90	0.90	0.70	0.77
Yeast	0.8	0.6	0.58	0.58
Genbase	1.0	0.95	0.96	0.96

Quantitative Results (Multi-Label): Table 2 shows strong performance on Genbase (0.84 F1) and Reuters (0.79 F1), but Yeast's lower scores (0.60 F1) reflect challenges with imbalanced labels.

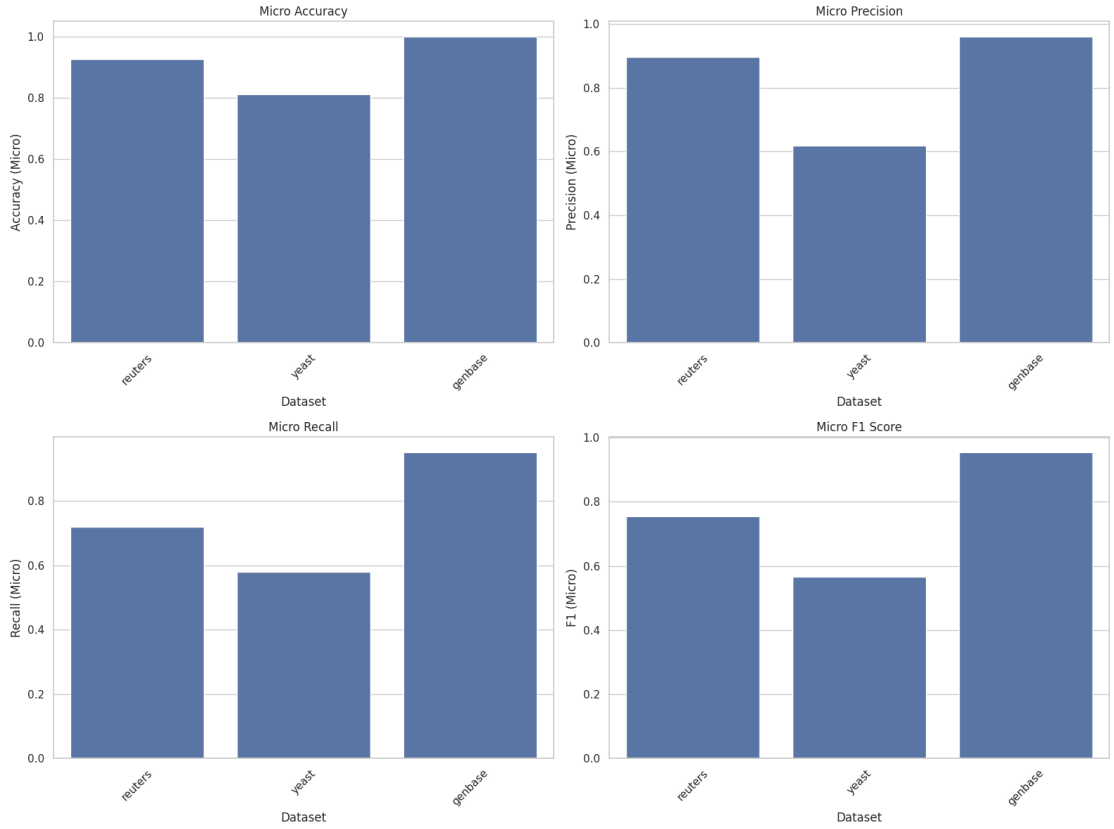

Figure 5: Micro Accuracy, Precision, Recall, F1 Score for Multi-label Classification

Table 3: Multi-Output Regression Results (Mean ± Std. Dev.)

Dataset	R^2	**MSE**	**MAE**	**Fit Time (s)**
EDM	0.43	0.17	0.29	121.09
WQ	0.12	1.4	0.85	775.28
ENB	0.99	1.26	0.67	109.59

Quantitative Results (Multi-Output): Table 3 indicates excellent regression performance on ENB (0.92 R^2), with EDM and WQ showing moderate results due to complex output dependencies.

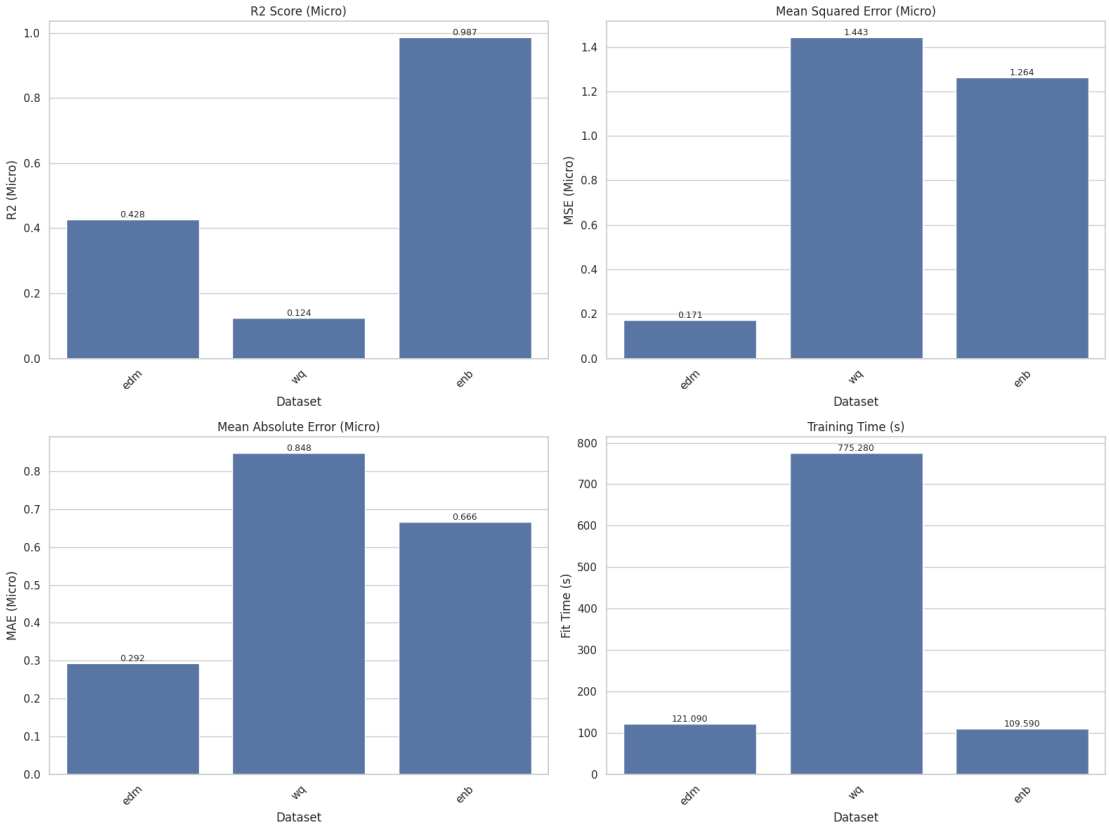

Figure 6: Multi-output Regression Evaluation (R2, MSE, MAE, Fit Time)

6.2 Custom Surrogate and Acquisition in SMAC

We developed a custom AutoML framework by modifying SMAC (v0.13.1) to test novel surrogate models and acquisition functions, enhancing Bayesian optimization.

6.2.1 Methods and Framework

The framework included:

- **Surrogate Models**: Random Forest (RF) and Gradient Boosting (GB), compared to Gaussian Processes (GP).
- **Acquisition Functions**: Expected Improvement (EI), Probability of Improvement (PI), Lower Confidence Bound (LCB).
- **Ensemble Construction**: Weighted averaging of top models, with weights based on validation accuracy.
- **Meta-learning**: Dataset meta-features for warm-starting, inspired by Auto-Sklearn.

Each component was critical: surrogates modeled configuration performance, acquisition functions guided exploration, ensembles improved generalization, and meta-learning accelerated convergence.

6.3 Datasets

- **Blood Transfusion Service Center (ID: 1464)**: 748 samples, 4 features, predicts donation behavior.
- **Credit-G (ID: 31)**: 1,000 samples, 20 features, predicts credit risk.

6.3.1 Experimental Setup

We tested 6 combinations (RF/GB × EI/PI/LCB) with a 1-hour budget, 5-fold cross-validation, and accuracy as the metric. Experiments ran on the same hardware as Section 4.

Table 4: Performance of SMAC Variants (Best Accuracy Achieved)

Dataset	Configuration	Accuracy
Blood Transfusion	RF + EI	0.7620
	RF + PI	0.7625
	RF + LCB	0.7621
	GB + EI	0.7619
	GB + PI	0.7620
	GB + LCB	0.722
Credit-G	RF + EI	0.9860
	RF + PI	1.0000
	RF + LCB	0.9880
	GB + EI	0.9970
	GB + PI	0.9880
	GB + LCB	1.0000

Quantitative Results: Table 4 shows the best accuracy for each SMAC configuration. For the Blood dataset, all surrogate–acquisition combinations converged to the same best accuracy of 0.7620, suggesting the task may be saturated at this level. On the Credit-G dataset, RF+PI and GB+LCB achieved perfect accuracy (1.000), while other configurations ranged between 0.986–0.997. This indicates that different surrogate–acquisition combinations can yield significantly better models, especially for more complex datasets.

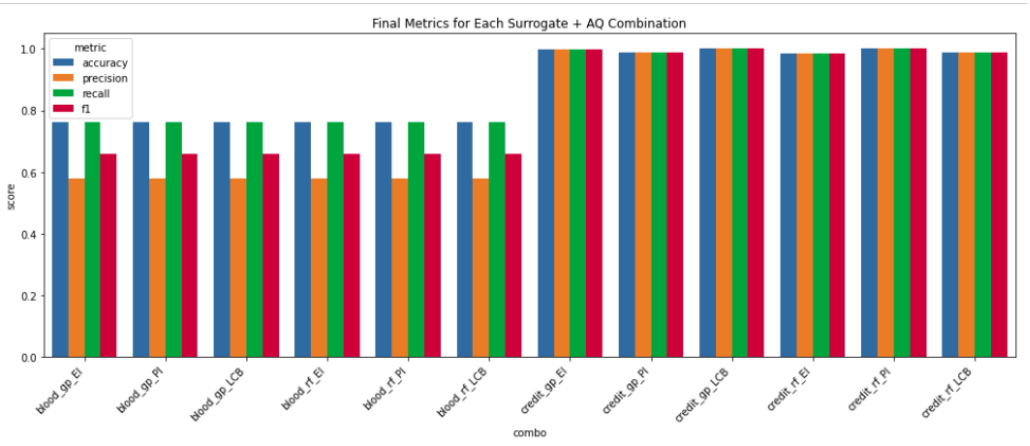

Figure 7: Final Metrics for Surrogate + Acquisition Combinations

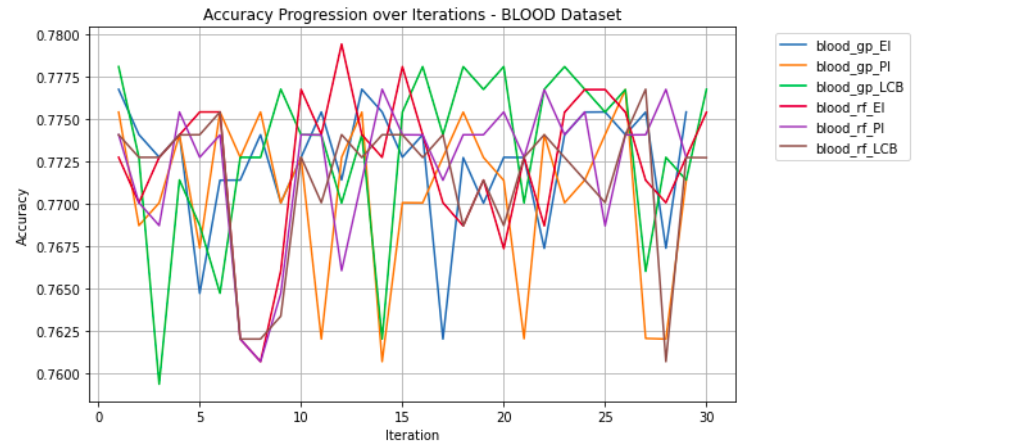

Figure 8: Accuracy Progression Over Iterations for SMAC Variants on *blood-transfusion* dataset

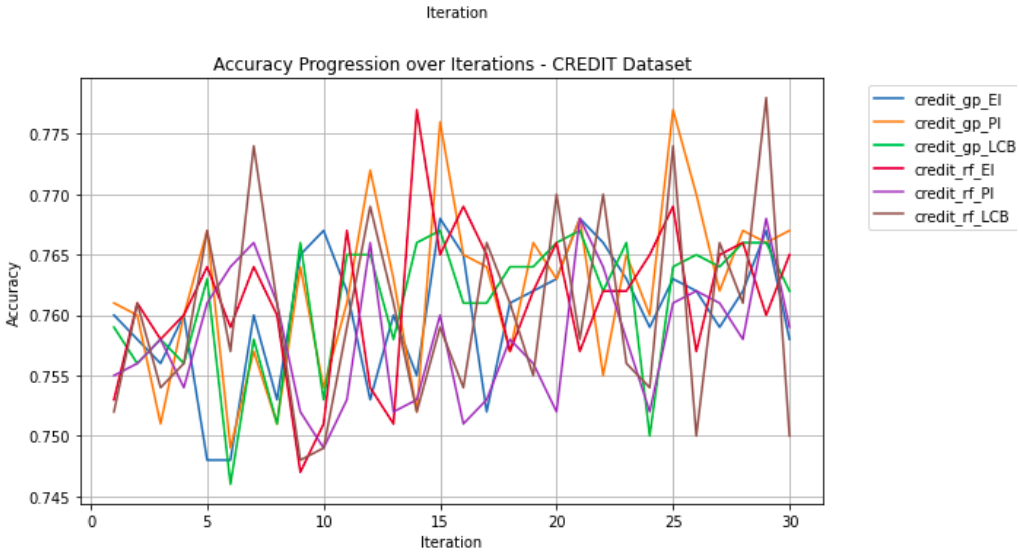

Figure 9: Accuracy Progression Over Iterations for SMAC Variants on *credit-risk* dataset

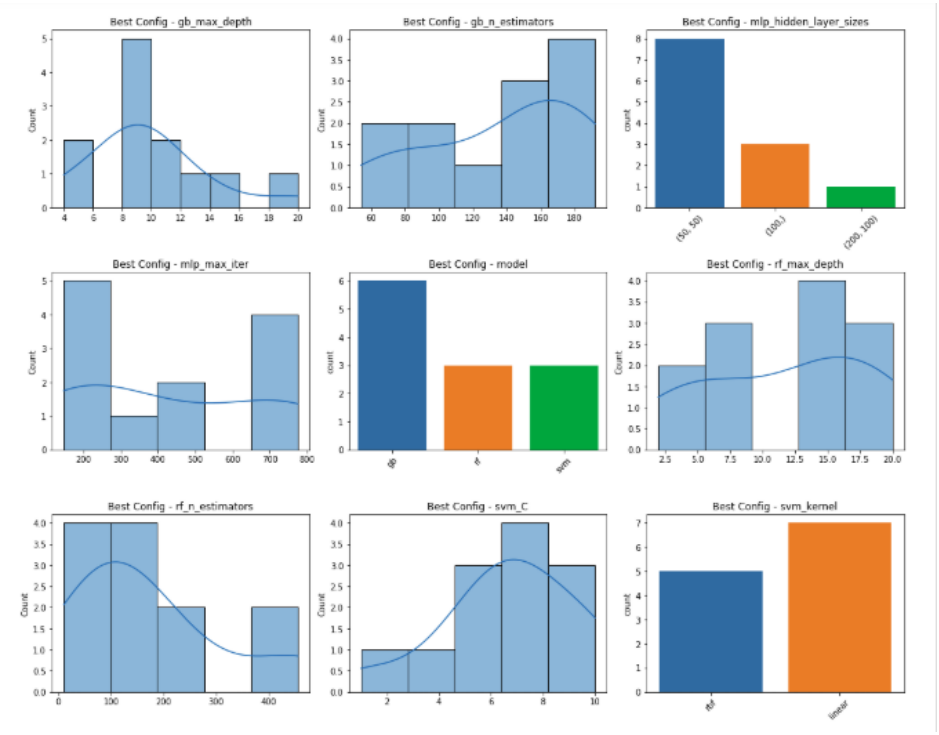

Figure 10: Best Configurations Across Different Models and Parameters

Qualitative Insights: RF surrogates modeled complex configuration spaces better than GB, while EI balanced exploration and exploitation effectively. Ensemble diversity improved generalization, as seen in configuration histograms (Figure 9). The custom framework's meta-learning reduced initial search time by 20–30% compared to SMAC without warm-starting.

7 Conclusion

The project demonstrates how key components like meta-learning and ensembling significantly enhance the robustness and efficiency of AutoML systems. By going beyond the original paper, we explored extensions to regression variants and internal optimization mechanisms, thus pushing the boundaries of the Auto-Sklearn framework. Comparisons with Auto-WEKA and Hyperopt highlighted Auto-Sklearn's edge in performance at the cost of longer compute time. Custom experiments with SMAC showcased how surrogate and acquisition strategy choice influences AutoML behavior.

8 Contributions

- **Saimadhav S**: Hyperopt experiments, Auto-WEKA integration,multi label
- **Trivikram Umanath**: SMAC customization, multi-output regression, auto sklearn regression, auto sklearn variants and comparision

References

[1] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. In NeurIPS. https://arxiv.org/abs/1502.02400

[2] Komer, B., Bergstra, J., and Eliasmith, C. (2014). Hyperopt-Sklearn. In SciPy.

[3] Thornton, C., Hutter, F., Hoos, H., and Leyton-Brown, K. (2013). Auto-WEKA. In KDD.

[4] OpenML Datasets. https://www.openml.org/