Universiteit Leiden
The Netherlands

Opleiding Informatica

Evaluating The Impact of Integrating Domain-Specific Assumptions Into Auto-sklearn

Tim Haasdijk

Supervisors: Laurens Arp & Mitra Baratchi

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

15/1/2025

Abstract

Automated Machine Learning (AutoML) systems streamline the machine learning pipeline but frequently overlook the potential benefits of incorporating domain-specific knowledge. This thesis explores the impact of incorporating domain-specific knowledge, conducting an experiment on integrating linear separability assumption into Auto-sklearn. Integration of this assumption is done by pruning the search space, whereafter we compare the efficiency and performance (accuracy, F1-score and precision) of the standard Auto-sklearn to that of the system with domain knowledge. Our experiment on the OpenML-CC18 benchmark demonstrated that domain knowledge integration can improve computational efficiency, reducing the runtimes by an average factor of 1.27 while achieving comparable performance. Our statistical test confirmed that significant efficiency gains were made without significant performance compromises. Moreover, domain knowledge integration is particularly beneficial in applications where computational resources are sparse. Our findings indicate that domain knowledge integration into AutoML can enhance search strategies, increase efficiency and maintain robust performance.

Keywords: Automated Machine Learning (AutoML), Auto-sklearn, Domain-Specific Knowledge, Linear Separability, Search Space Pruning, Benchmark.

Code: https://github.com/TimH149/AutoML-DomainKnowledge

Acknowledgments

I want to thank everyone who has helped and led me during this project.
First, I owe a special and profound thanks to Laurens Arp. Your support, detailed feedback, and encouragement have been exceptional. Despite your own demanding schedule and pressure associated with your dissertation, you dedicated time to offer me critical feedback –occasionally late into the evening– you have motivated to stay on track. Your commitment and optimism were crucial for the completion of this thesis.
I would also like to thank Mitra Baratchi for her guidance and for providing the space for me to develop my ideas. Your input has been valuable and I appreciate you guiding me through this project.
I would also like to express my gratitude to the ADA research group for cultivating an encouraging academic environment.
Finally, I want to thank my family and friends for their continuous support and encouragement. Over this process, your patience, understanding, and words of motivation have meant so much to me.

Contents

1 Introduction **1**
- 1.1 Problem statement and thesis goal . . . 1
- 1.2 Research questions . . . 2
- 1.3 Thesis overview . . . 2

2 Background and literature review **3**
- 2.1 Automated Machine Learning (AutoML) overview . . . 3
  - 2.1.1 Introduction to Auto-sklearn . . . 3
- 2.2 Search space . . . 5
- 2.3 Domain knowledge . . . 6
  - 2.3.1 Definition and importance of domain Knowledge . . . 6
  - 2.3.2 Domain knowledge in Automated Machine Learning . . . 6
- 2.4 Linear separability . . . 8

3 Methodology **9**
- 3.1 Defining an assumption . . . 9
- 3.2 Model integration . . . 10
- 3.3 Experiments . . . 11
- 3.4 Evaluation framework . . . 11
  - 3.4.1 Performance evaluation framework . . . 11
  - 3.4.2 Efficiency evaluation framework . . . 13

4 Experiments **14**
- 4.1 Linear separability . . . 14
- 4.2 Experimental setup . . . 15
  - 4.2.1 Dataset selection . . . 15
  - 4.2.2 Data preprocessing . . . 16
  - 4.2.3 Determining linear separability . . . 17
  - 4.2.4 Testing for significance . . . 17
- 4.3 Integrating non-linear separability assumption in Auto-sklearn . . . 18
- 4.4 Modified and standard Auto-sklearn model training . . . 19
  - 4.4.1 Parameter analysis . . . 20
  - 4.4.2 Ensuring convergence . . . 20

5 Results and analysis **21**
- 5.1 System performance comparison . . . 21
- 5.2 System efficiency comparison . . . 23
- 5.3 Autosklearn parameter effect on performance difference . . . 25
  - 5.3.1 Effect initial configurations via meta-learning . . . 26
  - 5.3.2 Effect runcount-limit . . . 27

6 Discussion **29**
6.1 Insights from performance and efficiency comparisons 29
6.2 Limitations of the assumption informed search 29
6.3 Implications for AutoML 29

7 Conclusion and future Work **30**
7.1 Summary of findings 30
7.2 Answering the research question 30
7.3 Directions for future Research 30

1 Introduction

In recent years, supervised machine learning algorithms have been applied in various fields. Machine learning is a subset of artificial intelligence that enables machines to learn from data automatically. ML models have demonstrated to be successful in solving numerous predictive tasks across different domains. For example, Machine Learning (ML) has been implemented in stock market prediction, achieving impressive cumulative returns of 340%, 185%, 371% and 360% on four stock market indices [1]. Furthermore, assistance of ML in medical diagnostics can significantly improve balanced accuracy (84.7% vs 67.2%) for predicting prostate cancer [2]. Additionally, Machine Learning models like random forest can be used to optimize investment strategies and identify key predictive factors [3], highlighting its success across diverse domains.
With these successes, there is an increasing demand for making machine learning tools more accessible to users without specialized machine learning knowledge. To address this need, Automated Machine Learning systems have emerged. Automated Machine Learning (AutoML) systems lower the technical barriers by automating complex processes such as model selection and hyperparameter tuning.

1.1 Problem statement and thesis goal

AutoML systems frequently neglect the potential advantages of including domain-specific knowledge (A) in the learning process. This oversight can result in;

1. **Suboptimal model performance**: Systems that do not leverage A may produce models that fail to account for domain-specific constraints or relationships.

2. **Inefficient search**: Integrating domain knowledge A can give valuable insights about what configurations are likely to produce optimal results.

Knowledge about data properties, such as data distribution, feature dependency or feature importance, can provide valuable insights that could enhance AutoML performance. Studies have shown that integrating domain knowledge can make AutoML systems more efficient [4], transparent [5], and can improve performance [6]. However, it is often uncertain how effectively AutoML systems converge to "correct" or optimal solutions without such guidance [7]. Addressing this uncertainty, we explore how integrating domain-specific knowledge A can refine the search space of AutoML S into a constrained search space S_{mod} defined as:

$$S_{mod} = S \setminus A$$

Additionally, we propose a method to evaluate the impact of integrating this knowledge
We will explore how domain knowledge can be leveraged to prune the search space. The primary goal of this thesis is to address whether integrating domain knowledge into AutoML can:

1. **Improving model performance**: We evaluate whether AutoML systems that integrate A leads to improved model quality, measured by accuracy (Acc), precision ($Prec$), and F1-score ($F1$):

$$E[M_{mod}] \geq E[M_{std}]$$

where M_{mod} and M_{std} represent the performance of the modified and standard AutoML systems, respectively.

2. **Enhancing efficiency**: We evaluate whether integrating A can reduce the computational runtime (T) of the AutoML process:

$$T_{mod} \leq T_{std}$$

   where T_{mod} and T_{std} represent the runtime of the modified and standard AutoML systems.

Additionally, we outline a framework for evaluating the impact of integrating domain-specific knowledge into AutoML, illustrating its application through an empirical experiment.

1.2 Research questions

This thesis explores the following research questions:

- Can integrating domain-specific knowledge improve the accuracy, F1-score and precision of AutoML?
- Can integrating domain-specific knowledge improve the efficiency of AutoML?

1.3 Thesis overview

This thesis examines the impact of incorporating domain-specific knowledge into Automated Machine Learning (AutoML), conducting experiments using the Auto-sklearn framework. In Section 2, we review the literature and provide background on AutoML, introducing domain knowledge and its applications. Then we will provide a method (Section 3) to evaluate the impact of integrating a domain-specific assumption or knowledge. An experiment is conducted in Section 4 on linear separability. The results compare the modified and standard systems, revealing significant efficiency gains without sacrificing performance. The discussion delves into the practical limitations of the experiment. We will conclude with a summary of our findings and recommendations for future work.

2 Background and literature review

This section provides the fundamental concepts and related work essential to understanding the research presented in this thesis. The first section will give an overview of Automated Machine Learning (AutoML) emphasizing its goals, challenges, and existing frameworks. The focus will shift to Auto-sklearn, as it is the framework used in this thesis. This section will mention key techniques of Auto-sklearn, such as meta-learning and ensemble learning, and its methodology for enhancing model performance. In addition, we will explore the role of domain knowledge in AutoML, highlight its potential to streamline the machine learning pipeline, reduce computational costs, and enhance model interpretability.

2.1 Automated Machine Learning (AutoML) overview

AutoML, as mentioned above, aims to automate the machine learning process. This process of developing a machine learning model consists of several stages, such as data preparation, model generation and model evaluation.
In the model generation stage, AutoML typically addresses the Combined Algorithm Selection and Hyperparameter optimization (CASH) problem. This problem refers to the simultaneous process of selecting the most suitable machine learning algorithm for the task (e.g., support vector machine, random forest, decision trees, etc.) and optimizing its hyperparameters to optimize performance metrics like accuracy or F1 score. One of the earliest AutoML frameworks, Auto-WEKA, automatically handled the CASH problem by searching over different WEKA machine learning algorithms [8] and configurations [9]. More recent AutoML frameworks, like Auto-sklearn, also incorporate performance-enhancing techniques, like meta-learning and ensemble learning, that can improve the final model performance [10].

2.1.1 Introduction to Auto-sklearn

Auto-sklearn is built upon the scikit-learn library, a popular Python package that facilitates the implementation of numerous machine learning methods for both supervised and unsupervised machine learning tasks[11]. Meta-learning leverages historical knowledge that is acquired from earlier runs to inform the model optimization process for unseen data. Meta-features provide descriptive, structural, and administrative information on a dataset. Simply put, metadata is data that describes other data, which can make it easier to understand and classify different types of data. A knowledge base with meta-features and the corresponding best-performing machine learning pipelines, along with meta-features of the new dataset, allow meta-learning to determine which configurations are most suitable for a given dataset [12].
Upon encountering a new dataset, Auto-sklearn starts to select models and configurations that have previously demonstrated high performance for similar datasets, followed by fine-tuning through Bayesian Optimization (BO) for further optimization [13, 14]. BO is a probabilistic model-based approach for optimizing functions that are expensive to evaluate, such as evaluating model configurations. Based on previously evaluated configurations, BO fits a probabilistic model to capture the relationship between ML configurations and their measured performance [14, 15]. This probabilistic model, sometimes referred to as the surrogate model, places a prior over the configurations, representing the expected performance of the configuration [15]. These priors are

then used to select the most promising configuration. The evaluation of this configuration is used to update the surrogate model, whereafter, priors are once again calculated, and the most promising configuration is evaluated. This process is repeated iteratively until a predefined stopping criterion is met.
Compared to other optimization methods, like random search and grid search, BO takes into account the results of previous evaluations. This enables Auto-sklearn to identify optimal or near-optimal models without requiring human intervention or domain knowledge. Meta-learning can be leveraged to warm-start the BO process by providing information about suitable configurations that have previously shown good performance on datasets with similar meta-features. This approach prematurely highlights the promising parts of the search space, leading to more efficient optimization by eliminating the need to start the search from scratch [12].
Furthermore, Auto-sklearn utilizes ensemble learning to boost performance by combining predictions from multiple models. The fundamental concept of ensembling is that combining diverse models can yield superior performance than relying on a single model. An ensemble model is a combination of multiple individual ML models. Studies have shown that ensembles often outperform individual models [16, 17, 18, 19]. Auto-sklearn builds a weighted ensemble, where each algorithm in the ensemble has a weight representing its relative contribution to the final prediction. The weights are determined using a greedy selection algorithm, where models are iteratively added to the ensemble in a way that maximizes the ensemble's performance on the validation set [20, 21]. Each time the greedy algorithm selects a model, the model's weight increases proportionally to the number of times it is selected during the ensembling process [20] Notably, models can be selected multiple times, allowing better-performing models to have a greater influence on the final prediction.
The competitive edge of an ensemble over individual models lies in its ability to balance bias and variance. especially when dealing with complex datasets [20]. Variance and bias are critical concepts in ML that help us understand a model's performance. Variance denotes the degree to which a model is affected by fluctuations in the dataset. Models with high variance often fit closely to specific patterns within the dataset, this makes them susceptible to the phenomenon called overfitting. Overfitting occurs when the model cannot generalize and fits too closely to the training data. Bias refers to the situation where a model systematically produces biased predictions by assuming an oversimplified model to make predictions, which often leads to underfitting as the model misses crucial patterns in the data.
Figure 1 shows a scenario where a model (left) is underfitting the data, the model does not capture the complexity of the data, because it assumes a model that is too simple. This model exhibits high bias because it systematically produces bad predictions by assuming an oversimplified model and low variance because it is not affected by fluctuations in the data. On the other hand, a proper balance between complexity and simplicity leads to a good fit and robust model. Robust models have accurate predictions on both the training data and unseen data. Overfitting (right) demonstrates the opposite problem of underfitting, overfitting occurs when the model becomes too complex and fits the training data too closely (excessively low bias and high variance). This causes poor generalization and low performance on unseen data.

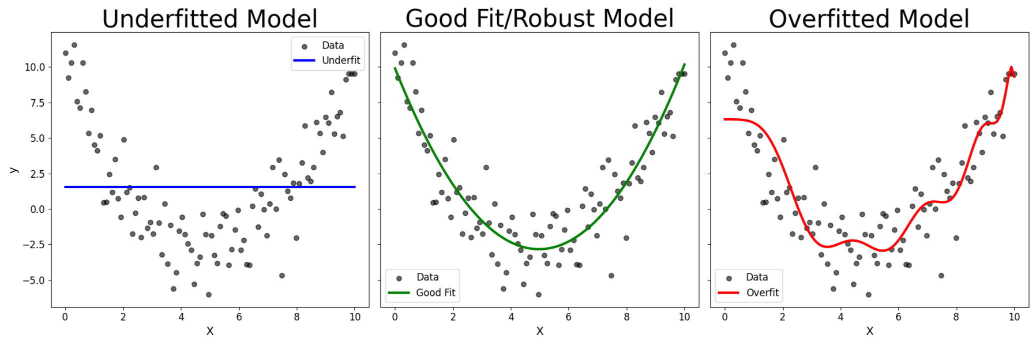


Figure 1: Comparison of underfitting, good fit, and overfitting This plot was created using synthetic data to visualize model-fitting behaviour.

Striking a balance between bias and variance is crucial for achieving a good fit. The bias-variance trade-off implies that too much bias leads to underfitting, while insufficient bias causes overfitting. Ensemble learning can address this trade-off by aggregating predictions, reducing the impact of errors from individual models, addressing both overfitting (high variance) and underfitting (high bias) [22]. For example, when a model overfits the data, ensembling multiple models can reduce variance by combining predictions from different models, averaging out variance errors. By averaging predictions, we reduce the likelihood that the error of a single model dominates the outcome, resulting in more stable and reliable predictions, increasing generalizability [23].

2.2 Search space

The search space of Automated Machine Learning can be referred to as all potential preprocessing techniques, machine learning algorithms, and their configurations that an AutoML system can consider while determining the most suitable model for a ML task. The search space consists of various components, like model types (e.g., decision trees, support vector machines, linear), preprocessing methods (e.g., normalization, feature selection), and hyperparameter values (e.g., learning rate, number of layers). The performance and efficiency of AutoML systems are strongly influenced by the complexity and size of this search space, as it determines how many configurations the systems must examine before arriving at the optimal model.
Efficiently navigating the search space is essential for making efficient and effective AutoML systems, as the number of configurations that the AutoML system can consider is enormous. Bayesian optimization is a key component that helps AutoML systems achieve this efficiency through an informed search strategy. Thornton et al. (2013) presented Auto-WEKA, a pioneering system that addressed the Combined Algorithms and Hyperparameter (CASH) optimization problem using Bayesian optimization methods [9]. Their work highlighted the need for efficient search strategies to make AutoML feasible for real-world applications. Auto-sklearn is built upon the ideas of Auto-Weka (Bayesian optimization and CASH) but extends it by incorporating additional optimization methods to efficiently explore the search space and increase performance.
As mentioned above, efficient exploration of the search space is crucial. Auto-sklearn makes use of advanced techniques such as meta-learning and BO to navigate the search space. Meta-learning helps identify which hyperparameters and models are likely to perform well on a new dataset based

on previously encountered datasets with similar data properties. By using this knowledge, the AutoML system starts with a more informed initial set of configurations [21]. Leveraging knowledge about promising models or the evaluations on the current dataset can significantly improve the efficiency and outcome of the search process. Another method to more efficiently navigate the search space is by pruning unpromising parts from the search space. Search space pruning is a search space reduction technique that removes specified configurations from the configuration space. Research has shown that introducing a pruning component to reduce the search space can increase search efficiency. For example, a pruning component that leveraged meta-learning to discard unpromising hyperparameter ranges significantly improved hyperparameter optimization efficiency [24].

2.3 Domain knowledge

2.3.1 Definition and importance of domain Knowledge

Domain knowledge refers to the specialized understanding of a specific field, industry, or area of study. In general, this understanding can provide contextual insights about the characteristics, concepts, and unique factors within the domain of application. For example, domain knowledge in the field of healthcare can be the understanding of medical terminology and treatment protocols. Domain knowledge in finance might involve understanding about stock or market trends.

2.3.2 Domain knowledge in Automated Machine Learning

Domain knowledge in Computer Science refers to prior, specialized expertise in a particular field besides event data [25]. Event data refers to all specialized expertise besides recorded sequences of occurrences, activities, or actions within a system or process. In Automated Machine Learning, domain knowledge can be leveraged to inform and guide the training process and data preparation phase [26] and enhance the interpretability of models [27]. Furthermore, domain knowledge can inform us about what configurations are suitable and unsuitable based on the characteristics of the domain. This knowledge can then be used to prune configurations from the configuration space, effectively narrowing the search space. This means that AutoML systems can focus resources on the most promising configuration, reducing the risk of bad fits. Examples of how domain knowledge can be integrated in AutoML:

- Feature engineering: We can leverage domain knowledge to find relevant features and relations between features within the data [28]. This information can then be used in the AutoML system to enhance their accuracy [29].
- Model Interpretation: Using domain knowledge, the models' prediction can be better interpreted as it can be put in the context of the domain, which increases model reliability [30].
- Data preprocessing and cleaning: Domain knowledge can inform us about abnormalities or noise within the data. This allows us to improve the overall quality of the data, which in turn can improve the model performance [31].
- Reduce computational costs: Domain knowledge can inform the AutoML system about the more promising parts of the search space. With this information, the AutoML system can focus on exploring the more promising part, possibly resulting in more efficient exploration and better model performance.

This thesis aims to explore the impact of domain knowledge integration in AutoML on computational costs and performance. Computational costs can be reduced by pruning the search space or feature space based on domain knowledge. By selectively narrowing the search space, AutoML systems like Auto-sklearn can reach high-performing models more quickly, potentially improving the final model performance and search efficiency of the system.
While prior research has shown that domain knowledge can successfully be leveraged to enhance model interpretation [27] and feature engineering [26], less attention has been given to search space pruning before initialization of BO. This thesis expands upon prior research by examining the impact of domain-informed pruning strategies on efficiency and performance. Methods for integrating domain knowledge to prune the search are:

- Pruning the search space through feature selection:

  Feature selection is defined as the process of identifying the relevant features in the input dataset. This process aims to identify the most informative or impactful features while removing redundant or irrelevant features. Because irrelevant features add unnecessary complexity and noise to the model, they can increase the risk of overfitting.

  In high-dimensional data, not all features contribute equally to the prediction. Removing redundant or irrelevant features reduces the complexity of the data. By focusing on a smaller, more relevant subset of features AutoML systems can avoid configurations within the search space that incorporate less informative features, which in turn can reduce the time that AutoML systems spend on searching the model and hyperparameter space.

  Standard automated feature selection tools that do not incorporate domain knowledge may struggle to determine the true value of a feature based solely on metrics. For example, in agriculture crop yield forecasting, a feature like "used tractor hours" may show a significant correlation with yield and be selected as a strong predictor for yield. However, this feature mainly reflects farm size and does not directly affect yield. In these cases, domain knowledge can play a critical role in enhancing feature selection, as it can give domain-specific insights about the relevance of features within the domain.

- Pruning the search space based on algorithm suitability within the domain:

  In some domains, certain algorithms consistently outperform others simply because the nature of the data aligns better with the characteristics of those algorithms. Olson et al [32] benchmarked 13 different scikit-learn algorithms on 165 datasets related to biomedical classification problems. The study showed that a random forest outperformed the other algorithms in terms of accuracy and generalizability. In domains where the interpretability of the model is crucial, the AutoML systems should prioritize models that are easy to understand and describe (e.g. linear model or decision tree).

  Domain knowledge can inform us about suitable models for a task in a domain, which can help narrow down the set of algorithms in the Auto-sklearn search space. This allows the system to concentrate computational resources on the most promising models. This approach could enable Auto-sklearn to reach optimal solutions more quickly and reduce the likelihood of overfitting or algorithm mismatch,possibly resulting in increased efficiency and performance.

- Pruning the search space through hyperparameter insights

  Exploring the search space for optimal hyperparameter values is a computationally extensive process. Domain knowledge can be leveraged to narrow the hyperparameter ranges that the system can consider to more realistic values. For example, in a classification task where the classes in the data are highly imbalanced, domain knowledge about the imbalance ratio within the domain can guide the choice of the minimum number of samples required to split a node. If the minority class is significantly underrepresented, domain knowledge might suggest that a higher minimum of samples per split can avoid splits that overfit to small subgroups of the majority class. With this knowledge, we can set a more appropriate maximum depth.

  Similarly, domain knowledge about the data's variability can help set an optimal learning rate. Lower learning rates allow the model to learn more gradually, avoiding overshooting and tend to be more suitable for domains with high noise.

  By leveraging domain knowledge to set more informed hyperparameter ranges, AutoML can avoid non-promising hyperparameter configurations. This approach can reduce the likelihood of over- or underfitting and reduces the computational costs.

2.4 Linear separability

Linear separability is a fundamental concept in machine learning and classification tasks. Linear separability in a dataset refers to the situation where we can separate the different classes of a dataset with a linear boundary, or in datasets with higher dimensions a hyperplane. Consider the following plots on two binary classification datasets:

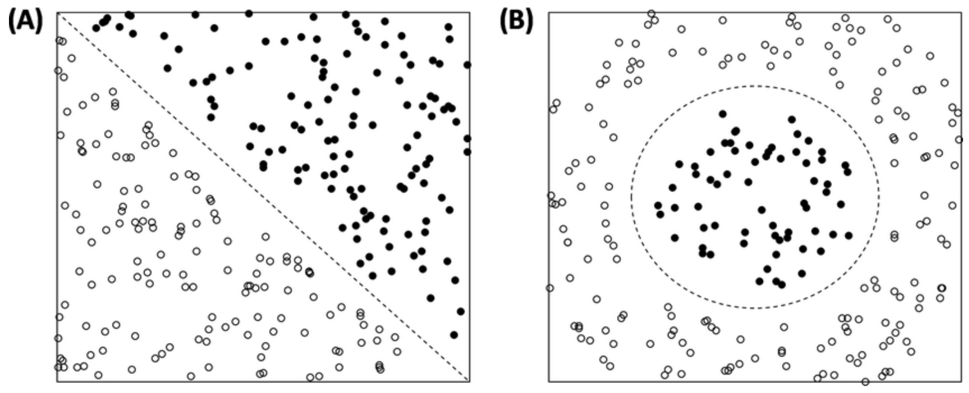


Figure 2: Linear separability vs non-linear separability. Image from Grove and Blinkhorn [33]

In this example, (A) shows two classes that can easily be separated using a linear boundary. Therefore, this dataset is linearly separable. (B) shows a more complex binary classification dataset that cannot be separated with a linear boundary, meaning that this dataset is non-linearly separable.

3 Methodology

We aim to explore whether the integration of domain-specific assumptions can improve the accuracy, F1-score, precision and efficiency of AutoML. The four main phases of the methodology consist of: data and assumption collection, model integration, experiments, and evaluation.

3.1 Defining an assumption

To explore whether integrating domain-specific assumptions can enhance AutoML performance, we first need to identify a relevant assumption A. Where A is defined as a constraint or knowledge about the dataset or domain. Section 2.3.2 outlined 3 example methodologies for integrating domain-specific assumptions. These 3 examples will be classified based on whether they constrain the search space S or feature space X of the AutoML system.

Mathematically, an assumption A is defined as the union of all domain-specific constraints that can be applied to either the feature space or the search space:

$$A = A_{\text{feature}} \cup A_{\text{search}},$$

Feature Assumptions (A_{feature}) Feature assumptions about the relevance of a feature can be used to constrain the feature space X. These assumptions can be defined as:

$$A_{\text{feature}} = \{x_i \in X : \text{Relevance}(x_i) \leq T\},$$

where:

- X: The original feature space.
- x_i: An individual feature in X
- $\text{Relevance}(x_i)$: A function that measures the relevancy of feature x_i.
- T: A feature relevance threshold based on domain knowledge

Search Space Assumptions (A_{search}) Search space assumptions are categorized into two groups: those constraining the selection of algorithms and those restricting the ranges of hyperparameters. Search space assumptions (A_{search}) constrain the search space S and are defined as:

$$A_{\text{search}} = A_{\text{algorithms}} \cup A_{\text{hyperparameters}}.$$

Where each component of A_{search} can be expressed as:

1. Algorithm suitability constraints: Constraints on the set of all possible algorithms M where:
$$A_{\text{algorithms}} = M \setminus M_{\text{suitable}},$$
and M_{suitable} is defined as the set of suitable algorithms based on domain knowledge.

2. Hyperparameter constraints: Constraints set on the hyperparameter space H_m of each algorithm m. Implementing hyperparameter constraints can prune unpromising or invalid hyperparameter configurations from the search space. These constraints are formally defined as:
$$A_{\text{hyperparameter}} = \bigcup_{m \in M} H_m \setminus H_{\text{promising}}.$$
where:
    - M is the set of all candidate algorithms.
    - H_m is defined as the hyperparameter space for algorithm m
    - $H_{\text{promising}}$ is a subset of H_m, containing all hyperparameter configurations that are seen as valid/promising based on domain knowledge.

These assumptions can be used to prune irrelevant features, unsuitable models, and unrealistic hyperparameter configurations out of the default AutoML search space. By pruning the default search space, we can evaluate the impact of integrating a domain-specific on efficiency, accuracy, F1-score and precision.

3.2 Model integration

In this step, the selected assumption is integrated into the AutoML system through modification of its search space or feature space. The integration of the domain assumption A into an AutoML system results in a modified system AutoML$_{\text{mod}}$. Mathematically described as:
$$\text{AutoML}_{\text{mod}} = \text{AutoML}(S_{\text{mod}}, X_{\text{mod}}, I),$$
where:

- S_{mod}: The modified search space.
- X_{mod}: The modified feature space.
- I: The target variable for which the AutoML is fitting a model (e.g., labels y).

Search Assumption (A_{search}) integration To integrate search assumptions, we apply the domain-specific search space constraints to the standard search space. These domain-specific search space constraints entail assumptions about the algorithm suitability and hyperparameter ranges, collectively referred to as A_{search}. The mathematical definition of pruning the standard search space S with domain assumption A_{search} to achieve a modified search space Smod can be defined as:
$$S_{\text{mod}} = S \setminus (A_{\text{search}}).$$

Feature Assumption ($A_{\textbf{feature}}$) integration To obtain the modified feature space X_{mod}, all irrelevant features A_{feature} are removed from the feature space (X). This is mathematically defined as:

$$X_{\text{mod}} = X \setminus A_{\text{feature}}.$$

3.3 Experiments

The experimentation phase evaluates the impact of integrating the domain-specific assumption on accuracy and efficiency. The experiments are designed to compare the modified system to the standard system:

- Baseline or standard model: The baseline model is generated by the standard version of Auto-sklearn; this model does not incorporate any domain-specific assumption. It is defined as:

$$\text{AutoML}_{\text{base}} = \text{AutoML}(S, X, I),$$

- The modified model is generated by the customized version of Auto-sklearn, which integrates a domain-specific assumption into its feature space or configuration space.

$$\text{AutoML}_{\text{mod}} = \text{AutoML}(S_{\text{mod}}, X_{\text{mod}}, I).$$

3.4 Evaluation framework

3.4.1 Performance evaluation framework

The next step is to evaluate the impact of the integrated assumption. To evaluate the performance of the modified and base system, we will use metrics such as precision, F1-score, accuracy, and recall. These metrics allow for a detailed comparison of the systems' effectiveness.

Notation Let K denote the number of classes in the dataset. For each class c, the following terms are defined:

- **True Positive (TP)c:** The number of instances that are correctly classified as class c.
- **False Positive (FP)c:** The number of instances that are classified as class c, but actually belong to another class.
- **False Negative (FN)c:** The number of instances that actually belong to class c, but are classified as another class.
- **True Negative (TN)c:** The number of instances that are correctly classified as not belonging to class c.

- Precision: Precision measures the proportion of correctly classified positive instances out of all instances that were classified as positive.

$$\text{Precision}_c = \frac{\text{True Positives (TP)}_c}{\text{True Positives (TP)}_c + \text{False Positives (FP)}_c}$$

  A high precision indicates that the model is minimizing the number of false positives.

- Recall: Recall quantifies what proportion of a class c is correctly classified as class c. It measures the model's ability to correctly identify all instances of a class.

$$\text{Recall}_c = \frac{\text{True Positives (TP)}_c}{\text{True Positives (TP)}_c + \text{False Negatives (FN)}_c}$$

  High recall indicates that the model successfully identifies most of the true positives while minimizing false negatives.

- F1 score: The F1-score is calculated as the harmonic mean of precision and recall. F1-score can give useful insights when there is an uneven class distribution.

$$F1_c = 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}$$

  A high F1-score suggests that there is a good balance between recall and precision.

- Accuracy: Accuracy measures the proportion of instances that were correctly classified out of all instances in the dataset.

$$\text{Accuracy} = \frac{\sum_{c=1}^{K} \text{TP}_c}{\text{Total Samples}}$$

  High accuracy means that a large proportion of the instances are correctly classified.

When evaluating the performance of multi-class classification tasks, it can be challenging to consider every metric for each class. Therefore, weighted averages are calculated for each metric. In our evaluation, weighted averages are preferred as they account for class imbalances in the dataset. The alternative, macro averaging, treats each class as equal without accounting for their size. However, this is crucial when dealing with class imbalances, which are often found in real-world situations. To calculate the weighted average of a metric, we must first assign a weight to each class. This weight is relative to the size of the class in the dataset. For a metric Metric:

$$\text{Weighted Metric} = \frac{\sum_{c=1}^{K} (\text{Metric}_c \cdot \text{Support}_c)}{\sum_{c=1}^{K} \text{Support}_c}$$

Where Support_c is the number of instances in class c.

3.4.2 Efficiency evaluation framework

To evaluate the computational efficiency of the systems, we compared the proportional runtimes (total_time_prop). This metric compares the runtimes of the two systems under the same dataset. This proportional metric can then be used to assess whether integrating our domain assumption has any significant impact on computational efficiency.

Calculating runtimes Runtimes will be calculated for the entire process of creating a model, this is to ensure that the analysis captures the real-world computational demands of using Auto-sklearn. The runtimes are calculated using the time.time() function from the Python time Module.

Calculating total_time_prop For each dataset, the runtime of the standard system (t_{standard}) was divided by the runtime of the modified system (t_{modified}). This was done to get a robust efficiency comparison across different datasets with varying complexity levels and sizes. It allows for a direct comparison and does not take absolute runtimes into account. total_time_prop is calculated using the following formula:

$$\text{total\_time\_prop} = \frac{t_{\text{standard}}}{t_{\text{modified}}}$$

This proportion represents how many times the standard system was faster or slower compared to the modified.

4 Experiments

This section focuses on experimentation of the methodology described in section 3. The domain-specific assumption that will be integrated is about linear separability in classification problems (see Section 2.4). Specifically, the assumption that every dataset provided to the AutoML system is non-linearly separable. This section will provide an example of integrating domain-specific knowledge into Auto-sklearn and describe how and whether the integration will have any impact on the system performance.

4.1 Linear separability

Most datasets are not linearly separable due to the inherent complexity of data in real-world applications. This complexity often involves non-linear patterns, for example:

- **Overlapping features:** Classes frequently overlap in real-world scenarios, rendering the establishment of a linear boundary to perfectly separate the classes unfeasible. For example, legitimate and fraudulent transactions often intersect in the feature space, as fraudulent activities are designed to mimic legitimate behaviour to avoid detection. In visual recognition, classes often have similar visual features (different breeds of dogs, facial recognition etc.).
- **High dimensionality:** In high-dimensional datasets, features are often interdependent rather than independent. Interdependency often results in complex relationships between feature and target variables that cannot be separated using a linear boundary.
- **Noise:** Real-world data is frequently noisy, with outliers and mislabelled instances further hindering the establishment of a linear boundary. For example, on a spam email classification problem, legitimate emails might occasionally be mislabelled as spam. In addition, typos or inconsistent language in emails could introduce outliers.

Focusing on the assumption of non-linear separability can offer several advantages for the design and optimization of AutoML systems:

- **Clear testing criteria:** Simple techniques can be used to determine whether a dataset is linearly separable, such as checking if an SVM with a linear kernel can achieve perfect or extremely high accuracy on the training data [34]. Another method to check for linear separability is with convex hulls [35].
- **Impact on search space:** When Auto-sklearn can assume that linear separability does not hold, the search space of the system can be reduced so that it contains only non-linear models.

Because of the factors mentioned above, we cannot assume that (real-world) datasets are linearly separable, meaning that we cannot rely solely on linear models for classification. Therefore, determining whether a dataset is linearly separable or not is essential for the selection of suitable algorithms and modelling approaches. Attempting to use linear models on datasets that are not linearly separable often leads to suboptimal results:

- **Underfitting:** Linear models fail to represent the non-linear relationships between the features and target.
- **Inefficient Search:** AutoML systems might waste computational resources on exploring linear models that are not suitable for the data.

Integrating the domain-specific assumption that the current dataset is linearly separable can guide the system to focus on non-linear models that are better suited for the task at hand. In essence, when we can assume that there is non-linearly separable data, we prune all linear models from the search space to reduce the Auto-sklearn search.

4.2 Experimental setup

This section details the experimental setup used to evaluate the integration of domain-specific assumptions, specifically focusing on non-linear separability in datasets, into Auto-sklearn.

4.2.1 Dataset selection

To ensure the generalizability of the proposed modification, we used a benchmark dataset from OpenML [36]. OpenML is a platform that hosts a wide range of datasets across various domains. This diversity allows us to compare the performance under different conditions, domains, class distributions, and feature relationships.
The dataset we used is OpenML-CC18 curated classification benchmark [37]. This benchmark dataset contains data from a wide variety of domains, allowing us to test the generalizability of the integration. This suite consists of 72 classification datasets that are carefully curated to fulfil the following inclusion criteria:

- Classification tasks on dense datasets with independent observations.
- Number of classes must be ≥ 2, with each class having at least 20 instances and the ratio of the minority to majority class exceeding 5
- Number of instances is between 500 and 100,000.
- After one-hot encoding, the number of features $< 5{,}000$.
- No dataset is artificial, a subset of larger datasets, or a binarization of another dataset.
- No dataset is perfectly predictable by a single feature or simple decision tree.

Some datasets from the benchmark were excluded based on specific criteria. Our experiment focuses on evaluating the impact of integrating non-linear separability assumption in Auto-sklearn on classification tasks, specifically on tabular datasets with well-defined features and manageable complexity. Below, we mention the datasets that were excluded and their reason:

- CIFAR_10 (OpenML ID: 40927): An image-based dataset with 3061 features and 600,000 instances. This dataset is disproportionately large compared to other datasets in the benchmark, making it difficult to evaluate alongside the other datasets.
- Devnagari-Script (OpenML ID: 40923): An image dataset based on handwritten digits, containing 1025 features and 92,000 instances. Like CIFAR-10, this dataset is disproportionally large.
- Analcatdata_dmft (OpenML ID: 469): This dataset was excluded because one of the classes was named "None", which is ambiguous. Handling this class would require manual preprocessing steps or, when handled automatically, could introduce unintended effects on other datasets.
- Fashion-MNIST (OpenML ID: 40996): An image-based dataset on Zalando's articles. Standard classification models cannot effectively utilize the correlations between imaged-based features (e.g. spatial relationships between pixels).
- Sick (OpenML ID: 38): Every instance in this dataset contains at least one missing value resulting in a total of 6064 missing values on 3772 instances.
- MNIST_784 (OpenML ID: 554): A disproportionately large image-based dataset (70,000 instances, 784 features).
- Isolet (OpenML ID: 300): A speech recognition dataset, with features that represent audio signals. Feature form differs significantly from typical tabular data. Moreover, the dataset is disproportionally large (7797 instances and 618 features)

4.2.2 Data preprocessing

Although Auto-sklearn has preprocessing steps, categorical values still need to be transformed into numerical values, and missing values should be addressed before Auto-sklearn can construct a machine-learning pipeline. This was done using the following preprocessing function:

Algorithm 1: Data preprocessing algorithm

```
Input: Unpreprocessed Dataset $X$ with numeric,categorical and missing values
Output: Preprocessed dataset $X$.
foreach column in $X$ do
    if column == 'object' then
        Transform object categorical values to numeric codes using Label Encoding;
    else if column == 'category' then
        Transform categories to numeric codes using Label Encoding;
    Missing value imputation on $X$ using K-Nearest Neighbors Imputation (KNN) with
      $k = 5$;
    return Preprocessed dataset $X$;
```

Missing values were imputed using the K-Nearest Neighbors (KNN) algorithm. Emmanuel et al [38] demonstrates that KNN can effectively fill missing values while preserving the characteristics of the data. KNN aims to preserve that imputed values align with the local structure of the dataset by predicting missing values based on similar data points. In contrast, simpler approaches like mean, median or mode imputation fail to leverage the patterns within the data, reducing the dataset's variance and increasing bias. Jadhav et al. [39] shows that KNN outperforms other imputation methods on Normalized Root Mean Square Error (RMSE). In this study, we will exclusively use tabular data, making KNN the recommended imputation method for numeric data [40] and the best method for categoric data [41].
After imputation, the datasets were split into training and test sets. For each dataset, the proportions of these splits are:

- **Training set**: 80% of the dataset
- **Test set**: 20% of the dataset

No additional validation split is required, as cross-validation is used by default in Auto-sklearn.

4.2.3 Determining linear separability

As mentioned in Section 4.1, we cannot assume that there is linear separability or non-linear separability in a dataset. Therefore, we must first define a method to determine whether a dataset is non-linearly separable. To achieve this, we employ a linear support vector machine (LSVM), a well-established algorithm that is specifically designed for classification data with linear separability. To account for data imperfections (such as noise or overlap) while evaluating linear separability, we will use a soft-margin (C = 1.0) LSVM approach. This soft-margin approach aims to find a hyperplane with the maximum margin while allowing some slack. Margin is defined as the distance between the separating hyperplane and the closest point from a class.
For each dataset, we looped over all classes and performed One versus All (OvA) classification. OvA is an approach where a binary classifier is trained for each unique class against all other classes [42]. In our case, a linear SVM is trained for each class to determine whether the dataset can be linearly separated for each class.
Once a linear SVM is fitted on the OvA training data, we calculate its accuracy on the test data. The assumption of non-linear separability is validated if the accuracy score achieved by the linear SVM is less than 0.5 for one of the classes. This threshold indicates that the linear SVM is performing no better than random guessing. Once this threshold is violated, we can assume that the dataset is non-linearly separable, meaning that we can modify Auto-sklearn to include our domain-specific assumption.

4.2.4 Testing for significance

To determine whether the observed performance metric scores and efficiency measures are statistically significant, we performed a Wilcoxon signed-rank test. This non-parametric test is a good statistical test for our experiment containing 43 datasets because it is robust to outliers, and does not make assumptions about the shape of the data.

4.3 Integrating non-linear separability assumption in Auto-sklearn

It is essential to understand the underlying framework and structure on the available algorithms of Auto-sklearn before we can integrate the assumption that there is no linear separability in the data. Moreover, each algorithm's ability to handle non-linear separability significantly influences its suitability for our task, as each dataset is not linearly separable. As mentioned in Section 4.1, using linear models on non-linearly separable data leads to underfitting models and can lead to Auto-sklearn wasting resources on inherently suboptimal results. Below, we listed all classification algorithms supported by Auto-sklearn, labelling them as linear or non-linear models using Scikit-learn official documentation and the associated references.

Algorithm	**Architecture**	**References**	**Algorithm Name**
`adaboost`	Non-linear	Freund et al. [43]	AdaBoost
`bernoulli_nb`	Non-linear	McCallum et al. [44]	Bernoulli Naive Bayes
`decision_tree`	Non-linear	Documentation [45]	Decision Tree
`extra_trees`	Non-linear	Breiman et al. [46]	Extremely Randomized Trees
`gaussian_nb`	Non-linear	Zhang et al. [47]	Gaussian Naive Bayes
`gradient_boosting`	Non-linear	Friedman et al. [48]	Gradient Boosting
`k_nearest_neighbors`	Non-linear	Goldberger et al. [49]	k-Nearest Neighbors
`lda`	Linear	Hastie et al. [50]	Linear Discriminant Analysis
`liblinear_svc`	Linear	Fan et al. [51]	Linear Support Vector Classification
`libsvm_svc`	Non-linear	Chang et al. [52]	Support Vector Classification (SVC)
`mlp`	Non-linear	Popescu et al. [53]	Multi-Layer Perceptron (MLP)
`multinomial_nb`	Non-linear	Rennie et al. [54]	Multinomial Naive Bayes
`passive_aggressive`	Linear	Crammer et al. [55]	Passive Aggressive Classifier
`qda`	Non-linear	Hastie et al. [50]	Quadratic Discriminant Analysis
`random_forest`	Non-linear	Breiman et al. [56]	Random Forest
`sgd`	Linear	LeCun et al. [57]	Stochastic Gradient Descent (SGD)

Table 1: Overview of all classification algorithms, their architecture, references, and corresponding names.

The table above highlights that 4 out of the 16 algorithms operate on the assumption that there is linear separability. Consequently these 4 algorithms are less suitable for our case, which is characterized by the assumption of non-linear separability. Therefore, the linear algorithms (lda, liblinear_svc, passive_aggressive and sgd) can be pruned from the search space, creating a modified version of Autosklearn that integrates the domain-specific assumption about non-linear separability.

4.4 Modified and standard Auto-sklearn model training

The standard and modified versions of Auto-sklearn are trained using the following arguments:

Parameter	**Explanation**
`runcount-limit`	Limits the total number of configurations tested during optimization. This parameter is set to 300, so a sufficient budget is provided to test each classifier thoroughly.
`memory_limit`	Restricts memory consumption of each configuration to 50000 MB, ensuring complex models are not limited by memory. This value essentially deactivates the memory constraint.
`initial_incumbent`	Determines how the optimization begins. "RANDOM" initializes the process with randomly chosen configurations instead of configurations that are generally well-suited for many datasets, stimulating diverse initial exploration.
`include`	All preprocessing steps are turned off for both the modified and unmodified version of Auto-sklearn, ensuring that no transformations will take place that may mask linear separability. This ensures that the focus will be on evaluating the abilities of the two versions of Auto-sklearn to handle non-linearly separable data. Enabling preprocessing could result in transformations where linear models perform better than non-linear models. The modified version only includes non-linear models, while the standard version includes all model types.
`per_run_time_limit`	This parameter cuts off configurations after a certain time, guaranteeing that poor-performing or exceptionally computationally expensive configurations do not dominate the total time budget. This parameter is set to 40 seconds to ensure thorough exploration, but also a cut-off after a certain timeframe.
`ensemble_kwargs= {"ensemble_size": 1}`	This parameter specifies the number of models in the ensemble. Ensembles are turned off for simplicity of the final model
`initial_configurations_ via_metalearning`	By setting this parameter to 0, we disable meta-learning, avoiding pre-learned patterns or configurations. Using this approach, we isolate the impact of the integration domain knowledge on the performance. As discussed in Brazdil et al. [15], meta-learning also incorporates linear separability, as proposed in [58]. To verify the effect of meta-learning we will do an additional experimentation in 5.3.1
`dataset_compression`	By setting this parameter to false we avoid any potential distortions that may affect non-linear separability. Disabling data compression ensures that Auto-sklearn uses raw data without linear separability.

Table 2: Table showing parameter setting of Auto-sklearn

4.4.1 Parameter analysis

In addition to evaluating the overall performance between the modified system and the standard system, we also looked at the impact of specific AutoML parameters on these outcomes in Section 5.3. The parameters which effect will be evaluated are:

- Meta-learning: For this evaluation, we enabled meta-learning by allowing a warm start of the optimization. This can be done by setting the initial_configuration_via_metalearning to 50.
- Evaluation budgets: The influence of varying evaluation budgets has been evaluated.

4.4.2 Ensuring convergence

To ensure that the computational budget allocated for this experiment (300 evaluations) is sufficient, we analysed the loss across all 300 evaluations for both systems on each dataset. Figure 3 illustrates that the loss of both systems stabilizes well before reaching the runcount-limit of 300.

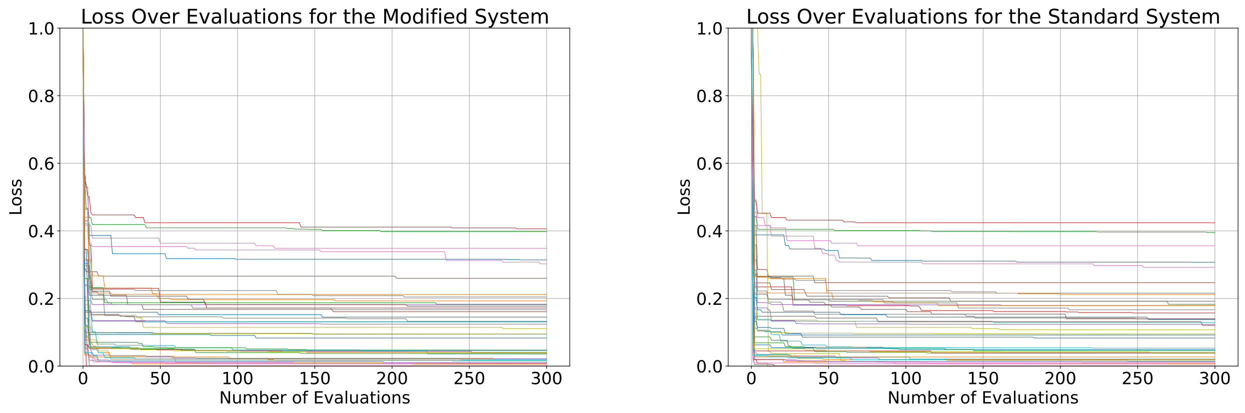


Figure 3: Loss (y-axis) over evaluations (x-axis) for the Modified (left) and Standard (right) system.

Figure 3 demonstrates that the loss experiences notable improvements during the initial evaluation phase, typically from 50 to 100 evaluations. After this phase, the systems' losses plateau, indicating convergence.

5 Results and analysis

This section presents the results and provides a comparative analysis of the performance and efficiency of the modified versus the standard version of Auto-sklearn. We used performance metrics mentioned in Section 3 to evaluate possible performance differences. In addition, for both systems, we used the time taken to converge to compare system efficiency. We tested for significance using a Wilcoxon signed rank test.

5.1 System performance comparison

To test whether integrating domain-specific knowledge can improve the accuracy, F1-score and precision of AutoML, we visualised the performance of both the modified and standard version in Figure 4. This plot is based on the accuracy results of the experiment, where each data point represents a "Dataset" with the accuracy score of the standard (x-axis) and modified system (y-axis). The red dashed line (y=x) indicates equal accuracy. Datapoints above the red dashed line represent datasets where the modified system outperformed the standard system, those below indicate datasets where the standard system was superior. Furthermore, in Table 3, we mention the mean and standard deviation of the difference between the modified and standard system (Modified - Standard) for each metric on a benchmark. The full accuracy table for each dataset and system can be found in the appendix Table 7.

Evaluations	**Accuracy Difference (Mean ± Std)**	**Precision Difference (Mean ± Std)**	**F1-Score Difference (Mean ± Std)**
300	0.00126 ± 0.0134	0.00105 ± 0.0199	0.000688 ± 0.0162
200	-0.00306 ± 0.0222	-0.00610 ± 0.0471	-0.00224 ± 0.02356

Table 3: Differences (Modified - Standard) in Accuracy, Precision, and F1-Score (Mean ± Std)

The mean differences between the modified and standard system for each metric are small, as shown by the low values in Table 3. This could suggest that the integration of domain knowledge has minimal impact on the performance metrics for the datasets tested. The standard deviations are relatively large in comparison to the mean differences. This suggests that there is a large variability in the performance difference. On some datasets, the modified system was superior, while on others, the standard system outperformed. We must perform a statistical test to determine whether the difference is significant. For our experimental setup, we perform a Wilcoxon signed-rank test to test whether there is a significant effect on the difference of a metric X. The hypotheses for the test were as follows:

- Null Hypothesis (H_0): The difference of metric X distribution is symmetric around zero.
- Alternative Hypothesis (H_a): The distribution of metric X differences between the two systems is not symmetric around zero.

The results of the test on each metric for each evaluation are mentioned in the table 4:

Evaluations	**Accuracy**		**Precision**		**F1-Score**	
	W	p-value	W	p-value	W	p-value
300	246.0	0.969	379.0	0.505	402.0	0.712
200	338.0	0.839	395.0	0.646	390.0	0.600

Table 4: Test statistic (W) and p-value for Accuracy, Precision, and F1-Score.

The outcome of the Wilcoxon signed-rank test indicates that the p-values for Accuracy, Precision and F1-score all exceed the significance level of 0.05. This signifies no statistical difference between the standard and modified system for these metrics. This suggests that integrating domain knowledge into Auto-sklearn does not significantly affect performance, using precision accuracy and accuracy scores to evaluate the performance between the systems with and without domain knowledge.

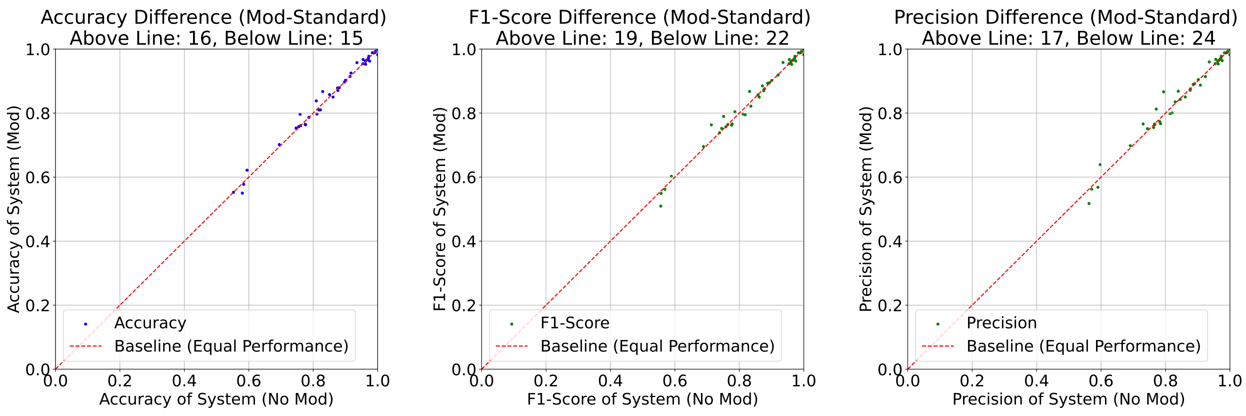


Figure 4: Scatter plot accuracy, F1 and precision scores of Modified vs Standard version.

These scatter plots compare the accuracy, F1 and precision scores of the modified Auto-sklearn (y-axis) to that of the standard version (x-axis). The scatter plots are based on the comparison results with 300 evaluations. The diagonal line represents the metric scores where both systems have equal scores. When analysing the accuracy plot we see that out of 43 datasets, the modified version outperformed the standard version 16 times, had equal performance 11 times, and had worse performance 15 times. This suggests that the modified system had the same or better performance 62.8 % of the time. For F1-score, the modified system achieved better scores on 19 datasets. However, on 22 datasets the modified system had worse performance. This indicates that on 51.2% of the datasets the modified system had better or equal performance. In terms of precision, the standard system more frequently achieved better results, with a count of 24. The modified system only performed better on 17 datasets. On 44.2% of the datasets the modified system had similar or better results. A large concentration of the points is near the top right of the plots, indicating that on many datasets both systems perform very well, for accuracy, precision, and F1-score. The modified system achieves comparable performance in most cases for most metrics. The statistical test combined with the scatter plot indicates that removing linear models from the standard system does not negatively impact the final model performance on these datasets.

5.2 System efficiency comparison

In this section, we compare the efficiency of the modified and standard system. We looked at the time taken for both systems to create a model. To test whether integrating domain-specific knowledge can improve efficiency, we performed a Wilcoxon test shown in Table 4. The proportional time is calculated as mentioned in 3.4.2. The histogram below (Figure 5), visualizes the distribution of the log-scaled total time proportional (total_time_prop).

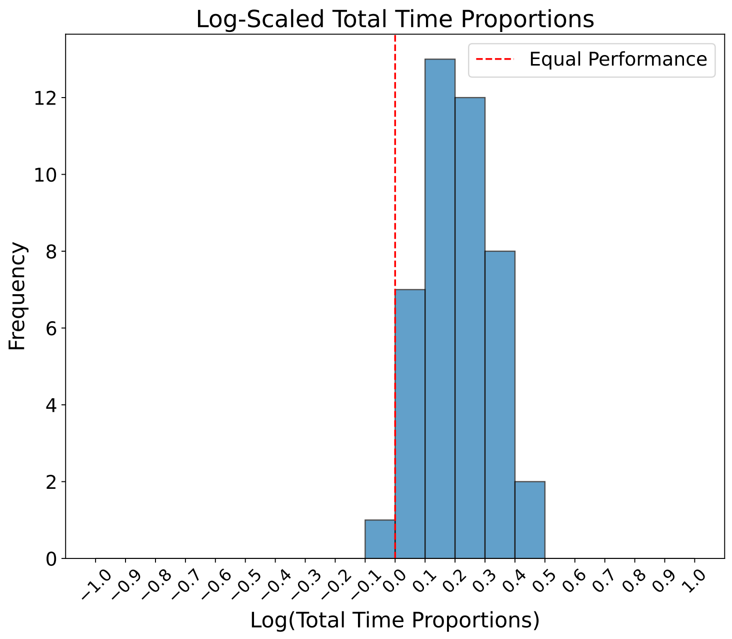


Figure 5: Log-scaled histogram of proportional time (Standard/Modified).

The figures show that the majority of the ratios cluster above 0, meaning that the modified system converged faster on the majority of the datasets. This highlights that excluding linear models from the search space when we can assume non-linear separability, is beneficial and saves time. Studies have shown that linear models are inherently faster [59, 60] due to their simplicity. With this knowledge in mind, one would expect minimal or no benefits in terms of efficiency. However, the results indicate that integrating non-linear separability assumption consistently improves computational efficiency.

The proportional time has an unimodal distribution, with the mode centring around 0.1-0.2 (normal time proportion: 1.11 - 1.22). This indicates that, in the majority of cases the modified system operates within slightly lower time ranges compared to the standard system. The largest time proportion was 0.47 (normal proportion 1.6), indicating that extreme time differences between the two systems were rare.

The mean of the log-scaled time proportions was 0.20, meaning that the average time proportion was:

$$e^{\text{mean}} = e^{0.20} \approx 1.27$$

On average, the modified system converged 1.27 times faster than the standard system.
To test whether there is a significant effect on the time efficiency we perform a Wilcoxon signed-rank test on the log-scaled time proportions. The hypotheses for the test were as follows:

- Null Hypothesis (H_0): The time proportion distribution is symmetric around zero.
- Alternative Hypothesis (H_a): The distribution of proportions between the two systems is not symmetric around zero.

The results of the test are as follows:

Evaluations	**Efficiency**	
	W	p-value
300	942.0	7.96e-13
200	920.0	1.22e-10

Table 5: Test statistic (W) and p-value for efficiency at different evaluations.

The extremely low p-values for these tests reveal that we can reject the null hypothesis. Additionally, it indicates that the system with domain knowledge is more efficient than the system without domain knowledge, suggesting a significant improvement in efficiency.

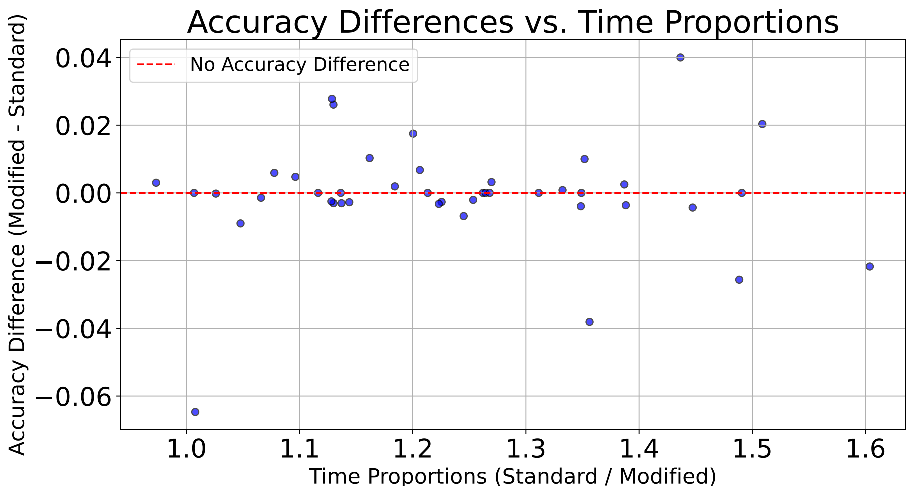


Figure 6: This figure illustrates the accuracy difference (Modified - Standard) on the y-axis against the proportional time differences on the x-axis for all selected, non-linear separable datasets in the benchmark.

Figure 6 shows that on the majority of the datasets, the accuracy score differences lie between -0.02 and 0.02, highlighting that the modified system achieves models with similar performance to the models found by the standard system. Moreover, the majority of the proportional time differences exceed 1 on the x-axis, indicating that the modified system achieves comparable result while requiring less computational time.

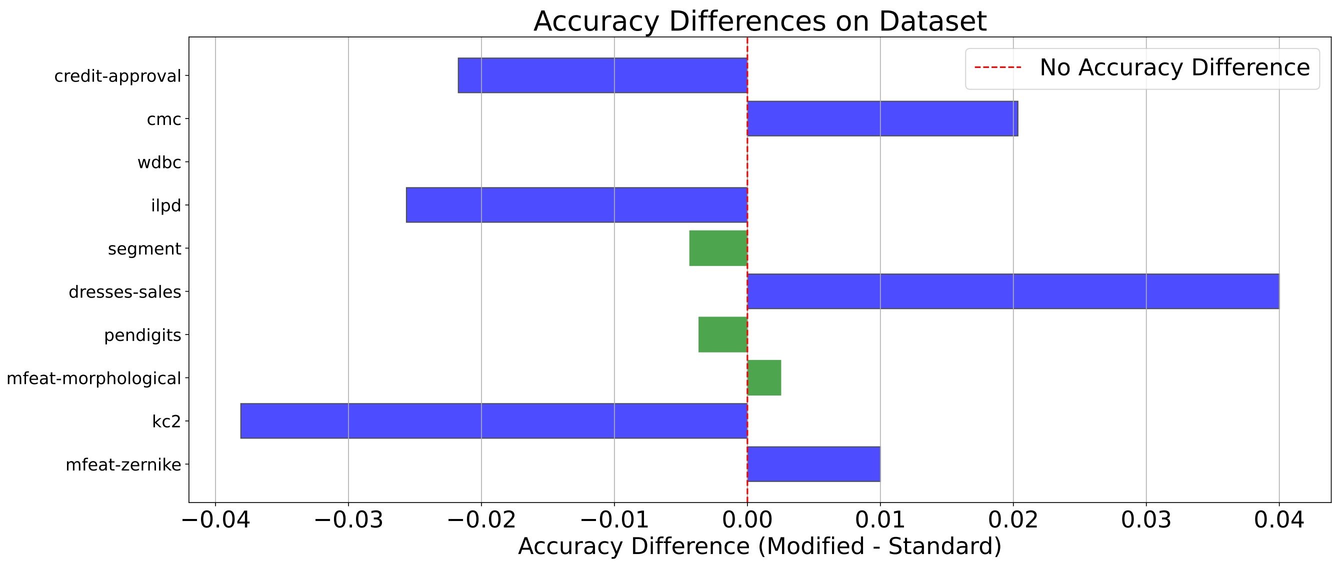


Figure 7: Modified vs Standard version where time proportion $\geq$1.35.

Figure 7 shows the accuracy differences between the modified and standard system (x-axis), and the dataset name (y-axis) for datasets where the standard system's runtime is at least 1.35 longer. This figure helps examine the accuracy differences where including linear models caused large time inefficiency.

For these datasets, the accuracy differences remain small (but relatively large compared to the differences at lower time proportions). There is no trend toward favouring one system over the other. The small accuracy differences, combined with the absence of a clear trend, highlight the trade-off: the modified system sacrifices negligible accuracy for significant time savings, making it highly practical for tasks where time efficiency is key.

5.3 Autosklearn parameter effect on performance difference

In this section, we evaluate the impact of meta-learning and lower evaluation budgets on the comparison.

5.3.1 Effect initial configurations via meta-learning

When meta-learning is enabled, even smaller performance differences between the modified and standard system are found. Below (Figure 6) we list the mean and standard deviation of the experiments with and without meta-learning. Meta-learning is enabled by setting the initial_configurations_via_metalearning to 50.

Meta-learning	**Accuracy Difference (Mean ± Std)**	**Precision Difference (Mean ± Std)**	**F1-Score Difference (Mean ± Std)**
Disabled	0.00126 ± 0.0134	0.00105 ± 0.0199	0.000688 ± 0.0162
Enabled	-0.000421 ± 0.0163	-0.00308 ± 0.0197	-0.000796 ± 0.0222

Table 6: Differences (Modified - Standard) in Accuracy, Precision, and F1-Score (Mean ± Std) with meta-learning and without meta-learning

Table 6 shows that the mean differences across all datasets come closer to 0 when meta-learning is enabled. This suggests that the small advantage of the modified system diminishes once meta-learning is enabled. Meta-learning leverages historical knowledge acquired from prior runs to inform the model optimization process for unseen data. Meta-learning allows Auto-sklearn to better navigate the search space by focusing on more promising parts of the search space 2.1.1. In essence, meta-learning performs a similar function to our domain knowledge integration without explicitly knowing whether the data is linearly separable: Based on metadata from previous experiments, meta-learning determines that non-linear models tend to outperform linear models for this certain type of data. As a result, when the initial_configurations_via_metalearning is set to 50 even smaller performance differences are found because the standard model with meta-learning applies the same underlying knowledge as the modified model.

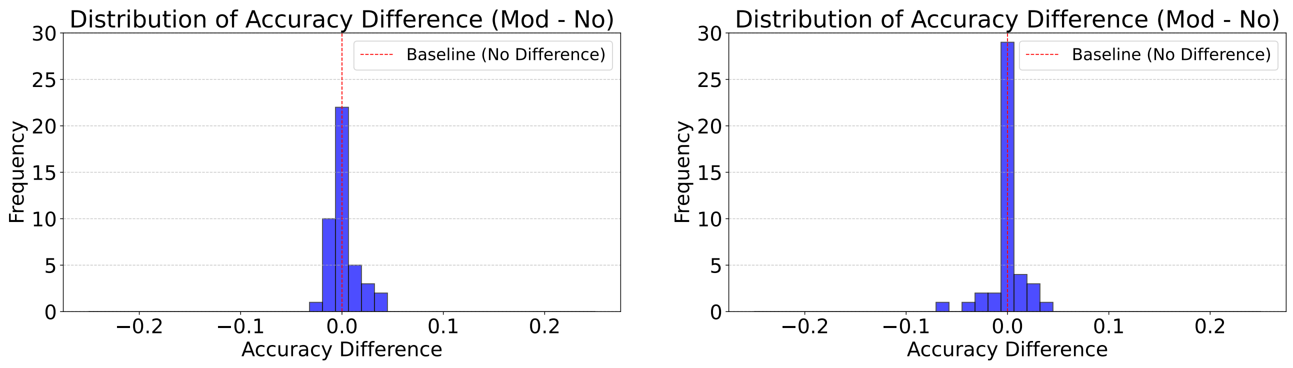


Figure 8: Plot showing accuracy difference distribution without (left) and with meta-learning (right)

Figure 8 shows that the accuracy difference distribution with and without meta-learning differs. The distribution with meta-learning (right) shifts closer to zero, while the distribution without meta-learning (left) has slightly more spread. In addition, accuracy differences are more frequently zero when meta-learning is enabled.

5.3.2 Effect runcount-limit

This parameter defines the maximum amount of evaluations allocated for Auto-sklearn to create a final model. As mentioned in section 5.1, the standard system produced models with comparable performance but, on average, required 1.27x as much time. Therefore, it is worth investigating whether lower budgets might favour the modified system by yielding better-performing models within the budget constraint.

To test this, we ran the experiment on different runcount-limits. In the original experiment, a runcount limit of 300 was used. To evaluate the effect of reducing the evaluation budget, we reduced the number of evaluations to 200, 100, 50, and at last we tried a very low budget of 15. The figure below (Figure 9) show the accuracy distribution for each evaluation budget.

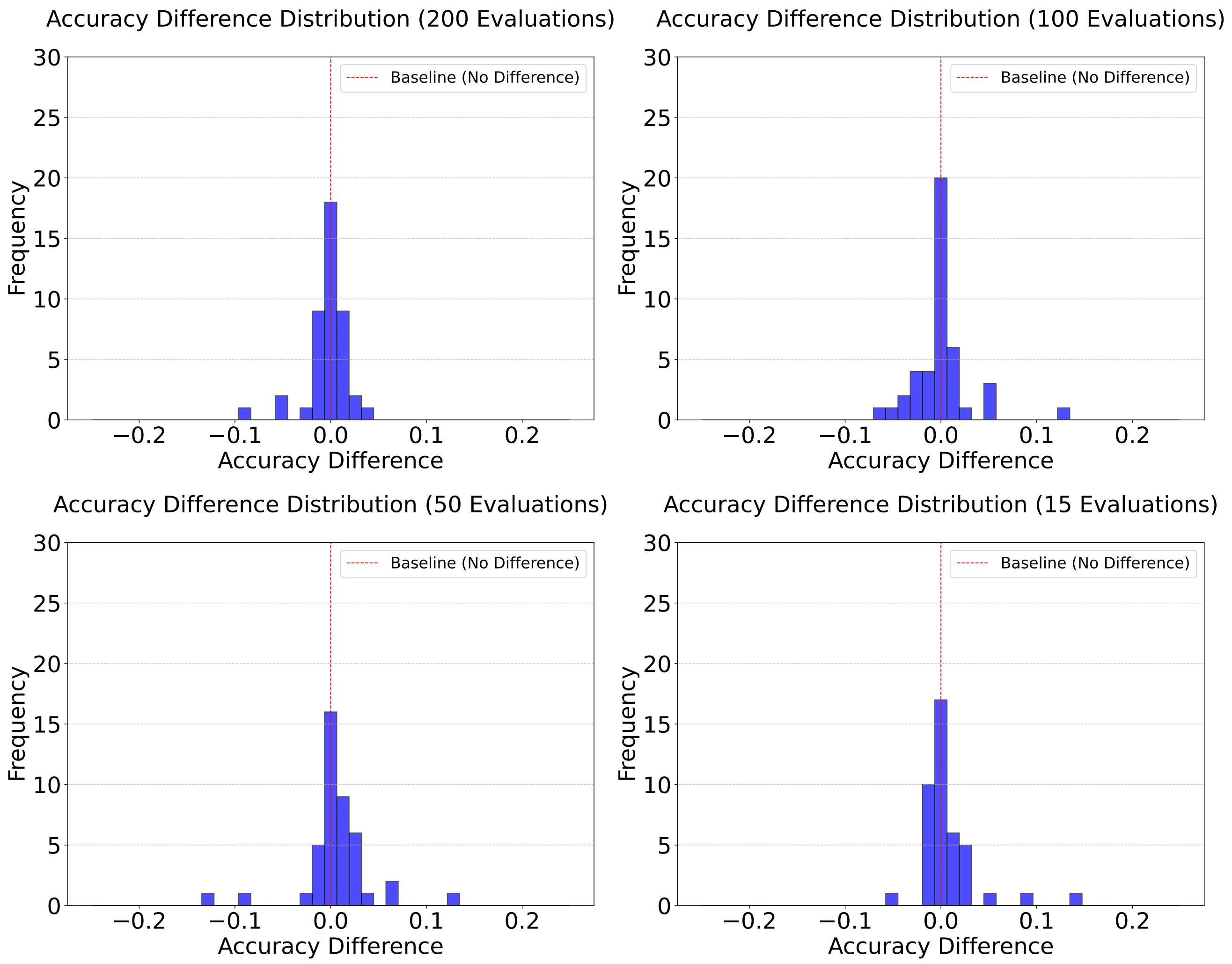
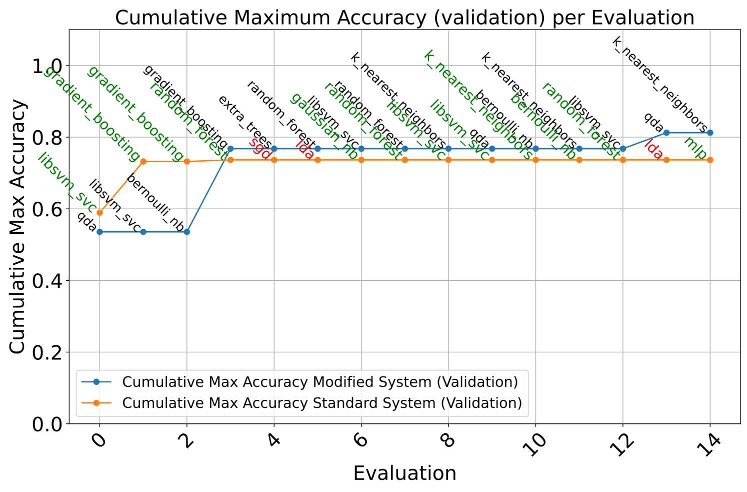


Figure 9: Accuracy difference distribution plot across varying evaluation budgets (200, 100, 50 and 15).

At the runcount-limits of 200 and 100 evaluations, both systems have comparable accuracy scores and distributions, with minor differences. The modified system demonstrates better performance relative to the standard system at the lower budgets. The effect of search space reduction is more noticeable as time constraints tighten since the modified system converges more quickly. The accuracy difference distribution of 15 evaluations (bottom-right plot) shows that the modified system achieved notable superior accuracy scores on 2 datasets. On these datasets (vehicle and cylinder-bands) the standard system spent 20% of its budget (3 evaluations) exploring linear models (sgd and lda), resulting in worse results. This is shown in Figure 10.



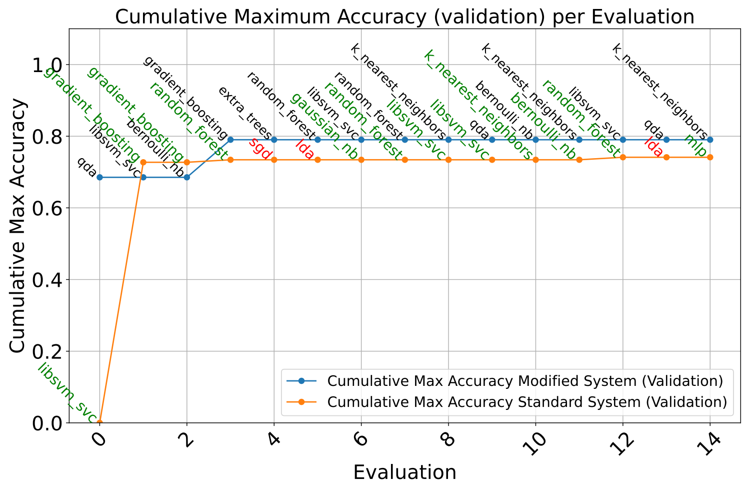


Figure 10: Cumulative accuracy score on validation set for datasets: vehicle (top) and cylinder(bottom)

In this figure, the orange line represents the standard system and the blue line represents the modified system. On the x-axis we plotted the evaluations, and the y-axis represents the cumulative maximum accuracy so far. The inability of linear models to capture the inherent complexities of these datasets, combined with the low evaluation budget, causes the standard system to waste valuable resources on configurations unsuitable for the data, resulting in lower accuracy scores.

6 Discussion

6.1 Insights from performance and efficiency comparisons

The findings on accuracy, F1-score, and precision indicate that the modified system performs slightly better or comparable on the majority of the datasets. When we can assume non-linear separability, pruning linear models from the search space does not significantly impact accuracy. The modified system consistently showed improved efficiency by reducing runtime without a significant accuracy trade-off.

6.2 Limitations of the assumption informed search

Despite the modified system achieving higher efficiency with comparable performance, there are some limitations to its applicability and generalizability. The primary limitation is that the modified system relies on the assumption of non-linear separability, which did not hold for all datasets in the benchmark. In these cases, excluding linear models can cause unnecessary constrain on the search space, resulting in suboptimal models (e.g. a non-linear model overfitting on linear data).
Furthermore, the experiment was conducted under specific conditions. Meta-learning and ensemble learning were intentionally excluded to isolate the impact of the domain-specific assumption. However, in real-world applications of Auto-sklearn, meta-learning and ensemble learning are commonly enabled to increase efficiency and performance. Therefore, this absence during the experiment limits the generalizability of the findings to practical settings where they are typically enabled.
Additionally, all preprocessing techniques were disabled to ensure that the raw data remained non-linearly separable. Transformations such as Principal Component Analysis (PCA) needed to be excluded as they could transform the data to linearly separable data.

6.3 Implications for AutoML

The findings illustrate the potential of integrating domain-specific knowledge to improve AutoML efficiency. Highlighting that focusing on more promising parts of the search space results in more effective use of computational resources. Domain knowledge integration can complement existing optimization strategies and open the door for a more targeted approach.

7 Conclusion and future Work

7.1 Summary of findings

In this thesis, we evaluated the impact of incorporating domain-specific knowledge into AutoML, particularly Auto-sklearn, on computational efficiency and model performance. To explore this, we conducted an experiment using our methodology for integrating domain-specific knowledge. This experiment utilized the premise of non-linear separability using a selection of datasets from the OpenML-CC18 benchmark. To create the modified system, the search space was pruned to only include non-linear models.
The integration of non-linear separability into Auto-sklearn reduced runtime by an average factor of 1.27 while yielding no significant difference in accuracy score. These results confirm that domain-specific knowledge, such as assumptions, can optimize AutoML processes. Our findings highlight the potential for enhancing AutoML through domain-knowledge-informed search, particularly in resource-constrained environments.

7.2 Answering the research question

- Can integrating domain-specific knowledge improve the accuracy, F1-score and precision of AutoML? No, in our experiment we found no statistically significant effect on performance metrics accuracy, F1-score and precision.
- Can integrating domain-specific knowledge improve the efficiency of AutoML?
  Yes, incorporating assumptions like non-linear separability significantly improves efficiency while maintaining comparable performance.

7.3 Directions for future Research

Future work can explore the following:

- Evaluate the framework on datasets with greater complexity: Further validation can focus on datasets with different levels of data complexity, such as dimensionality, data size, and noise. Evaluating the performance and efficiency of data with high complexity, along with setting stricter time limits, could reveal if the modified system outperforms the standard system in more challenging scenarios.
- Explore multiple domain-specific assumptions: The main focus of our work was on integrating the assumption about non-linear separability in data, but other domain-specific knowledge could also improve AutoML performance:
  - Multicollinearity: Knowledge about correlations between features could potentially guide the feature selection, model selection, and regularization process.
  - Class imbalance: Knowledge about class imbalances, such as when one class is significantly under-represented, could inform techniques such as oversampling or customized loss functions.

References

[1] Chalvatzis, Chariton and Hristu-Varsakelis, Dimitrios. High-performance stock index trading: making effective use of a deep LSTM neural network. 2019. arXiv: 1902.03125 [q-fin.ST]. URL: https://arxiv.org/abs/1902.03125.

[2] Mota, Sakina Mohammed et al. "Artificial intelligence improves the ability of physicians to identify prostate cancer extent". In: The Journal of Urology 212.1 (June 2024), pp. 52–62. DOI: 10.1097/ju.0000000000003960. URL: https://doi.org/10.1097/ju.0000000000003960.

[3] Kaczmaczyk, Klaudia and Hernes, Marcin. "Financial decisions support using the supervised learning method based on random forests". In: Procedia Computer Science 176 (Jan. 2020), pp. 2802–2811. DOI: 10.1016/j.procs.2020.09.276.

[4] Petersen, Brenden K., Santiago, Claudio P., and Landajuela, Mikel. Incorporating domain knowledge into neural-guided search. July 2021. URL: https://arxiv.org/abs/2107.09182?.

[5] Cofaru, Corneliu and Loeckx, Johan. A knowledge-driven AutoML architecture. Nov. 2023. URL: https://arxiv.org/abs/2311.17124?.

[6] Hollmann, Noah, Müller, Samuel, and Hutter, Frank. Large Language models for Automated Data science: Introducing CAAFE for Context-Aware Automated Feature Engineering. May 2023. URL: https://arxiv.org/abs/2305.03403?.

[7] Sun, Yuan et al. "AutoML in The Wild: Obstacles, Workarounds, and Expectations". In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI '23. ACM, Apr. 2023, pp. 1–15. DOI: 10.1145/3544548.3581082. URL: http://dx.doi.org/10.1145/3544548.3581082.

[8] Hall, Mark et al. "The WEKA data mining software: an update". In: SIGKDD Explor. Newsl. 11.1 (Nov. 2009), pp. 10–18. ISSN: 1931-0145. DOI: 10.1145/1656274.1656278. URL: https://doi.org/10.1145/1656274.1656278.

[9] Thornton, Chris et al. "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*. ACM. 2013, pp. 847–855. DOI: 10.1145/2487575.2487629.

[10] He, Xin, Zhao, Kaiyong, and Chu, Xiaowen. "AutoML: A survey of the state-of-the-art". In: arXiv preprint arXiv:2007.04074 (2020). URL: https://arxiv.org/abs/2007.04074.

[11] Pedregosa, Fabian and Varoquaux. "SciKit-Learn: Machine Learning in Python". In: vol. 12. Nov. 2011, pp. 2825–2830. URL: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?ref=https:/.

[12] Eldeeb, Hassan et al. "The impact of Auto-Sklearn's Learning Settings: Meta-learning, Ensembling, Time Budget, and Search Space Size". In: Mar. 2021.

[13] Feurer, Matthias et al. Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning. Tech. rep. 22, pp. 1–61. URL: https://www.jmlr.org/papers/volume23/21-0992/21-0992.pdf.

[14] Feurer, Matthias et al. "Auto-sklearn: Efficient and Robust Automated Machine Learning". In: May 2019, pp. 113–134. ISBN: 978-3-030-05317-8. DOI: 10.1007/978-3-030-05318-5_6.

[15] Brazdil, Pavel et al. Metalearning. Nov. 2008. DOI: 10.1007/978-3-540-73263-1. URL: https://doi.org/10.1007/978-3-540-73263-1.

[16] Hasrod, Taskeen, Nuapia, Yannick B., and Tutu, Hlanganani. "Comparison of individual and ensemble machine learning models for prediction of sulphate levels in untreated and treated Acid Mine Drainage". In: Environmental Monitoring and Assessment 196.4 (Mar. 2024). DOI: 10.1007/s10661-024-12467-8. URL: https://doi.org/10.1007/s10661-024-12467-8.

[17] Prusa, Joseph, Khoshgoftaar, Taghi M, and Dittman, Daivd J. "Using ensemble learners to improve classifier performance on tweet sentiment data". In: *2015 IEEE international conference on information reuse and integration*. IEEE. 2015, pp. 252–257.

[18] Anwar, Hina, Qamar, Usman, and Muzaffar Qureshi, Abdul Wahab. "Global optimization ensemble model for classification methods". In: The Scientific World Journal 2014.1 (2014), p. 313164.

[19] Lacoste, Alexandre et al. "Agnostic Bayesian learning of ensembles". In: 31st International Conference on Machine Learning, ICML 2014 2 (Jan. 2014), pp. 934–943.

[20] Caruana, Rich et al. "Ensemble Selection from Libraries of Models". In: July 2004. DOI: 10.1145/1015330.1015432.

[21] Feurer, Matthias et al. Efficient and Robust Automated Machine Learning. Ed. by C. Cortes et al. 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf.

[22] Maclin, Richard and Opitz, David W. "Popular Ensemble Methods: An Empirical Study". In: CoRR abs/1106.0257 (2011). arXiv: 1106.0257. URL: http://arxiv.org/abs/1106.0257.

[23] Kohavi, Ron and Wolpert, David H. "Bias Plus Variance Decomposition for Zero-One Loss Functions". In: Machine Learning: Proceedings of the Thirteenth International Conference (1997).

[24] Wistuba, Martin, Schilling, Nicolas, and Schmidt-Thieme, Lars. Hyperparameter Search Space pruning – a new component for sequential Model-Based Hyperparameter Optimization. Jan. 2015, pp. 104–119. DOI: 10.1007/978-3-319-23525-7\{_}7. URL: https://doi.org/10.1007/978-3-319-23525-7_7.

[25] Schuster, Daniel, van Zelst, Sebastiaan J., and van der Aalst, Wil M.P. "Utilizing domain knowledge in data-driven process discovery: A literature review". In: Computers in Industry 137 (2022), p. 103612. ISSN: 0166-3615. DOI: https://doi.org/10.1016/j.compind.2022.103612. URL: https://www.sciencedirect.com/science/article/pii/S0166361522000070.

[26] Soto, Patricia Centeno et al. "An ontology-based approach for preprocessing in machine learning". In: *2021 IEEE 25th International Conference on Intelligent Engineering Systems (INES)*. 2021, pp. 000133–000138. DOI: 10.1109/INES52918.2021.9512899.

[27] Islam, Sheikh Rabiul et al. Infusing domain knowledge in AI-based "black box" models for better explainability with application in bankruptcy prediction. 2019. arXiv: 1905.11474 [cs.AI]. URL: https://arxiv.org/abs/1905.11474.

[28] Kerrigan, Daniel, Hullman, Jessica, and Bertini, Enrico. "A survey of Domain Knowledge Elicitation in Applied Machine Learning". In: Multimodal Technologies and Interaction 5.12 (Nov. 2021), p. 73. DOI: 10.3390/mti5120073. URL: https://www.mdpi.com/2414-4088/5/12/73.

[29] Sharma, Shubham, Nayak, Richi, and Bhaskar, Ashish. "Multi-view feature engineering for day-to-day joint clustering of multiple traffic datasets". In: Transportation Research Part C: Emerging Technologies (2024). Queensland University of Technology, Australia.

[30] Huynh, Phat K. Knowledge integration in Domain-Informed machine learning and multi-scale modeling of nonlinear dynamics in complex systems. URL: https://digitalcommons.usf.edu/etd/10052/.

[31] Elouataoui, Widad, Mendili, Saida El, and Gahi, Youssef. An automated big data quality anomaly correction framework using predictive analysis. Dec. 2023. DOI: 10.3390/data8120182. URL: https://www.mdpi.com/2306-5729/8/12/182.

[32] Olson, Randal S et al. Data-driven advice for applying machine learning to bioinformatics problems. 2018. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC5890912/.

[33] Grove, Matt and Blinkhorn, James. "Neural networks differentiate between Middle and Later Stone Age lithic assemblages in eastern Africa". In: PLoS ONE 15.8 (Aug. 2020), e0237528. DOI: 10.1371/journal.pone.0237528. URL: https://doi.org/10.1371/journal.pone.0237528.

[34] Support vector machines: The linearly separable case. URL: https://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-the-linearly-separable-case-1.html.

[35] "Convex hull test of the linear separability hypothesis in visual search". In: CORE Reader (). URL: https://core.ac.uk/reader/82407420.

[36] Vanschoren, Joaquin et al. "OpenML: Networked Science in Machine Learning". In: SIGKDD Explorations 15.2 (2013), pp. 49–60. DOI: 10.1145/2641190.2641198. URL: http://doi.acm.org/10.1145/2641190.2641198.

[37] Bischl, Bernd et al. "OpenML Benchmarking Suites". In: arXiv:1708.03731v2 [stat.ML] (2019).

[38] Emmanuel, Tlamelo et al. "A survey on missing data in machine learning". In: Journal Of Big Data 8.1 (Oct. 2021). DOI: 10.1186/s40537-021-00516-9. URL: https://doi.org/10.1186/s40537-021-00516-9.

[39] Jadhav, Anil, Pramod, Dhanya, and Ramanathan, Krishnan. "Comparison of performance of data imputation methods for numeric dataset". In: Applied Artificial Intelligence 33.10 (July 2019), pp. 913–933. DOI: 10.1080/08839514.2019.1637138. URL: https://doi.org/10.1080/08839514.2019.1637138.

[40] Lalande, Florian and Doya, Kenji. Numerical Data Imputation: Choose kNN over Deep Learning. Jan. 2022, pp. 3–10. DOI: 10.1007/978-3-031-17849-8\{_}1. URL: https://doi.org/10.1007/978-3-031-17849-8_1.

[41] Memon, Shaheen MZ., Wamala, Robert, and Kabano, Ignace H. "A comparison of imputation methods for categorical data". In: Informatics in Medicine Unlocked 42 (2023), p. 101382. ISSN: 2352-9148. DOI: https://doi.org/10.1016/j.imu.2023.101382. URL: https://www.sciencedirect.com/science/article/pii/S2352914823002289.

[42] Gao, Xin et al. "A multiclass classification using one-versus-all approach with the differential partition sampling ensemble". In: Engineering Applications of Artificial Intelligence 97 (2021), p. 104034. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2020.104034. URL: https://www.sciencedirect.com/science/article/pii/S0952197620303110.

[43] Freund, Yoav and Schapire, Robert E. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: Journal of Computer and System Sciences 55.1 (1997), pp. 119–139. ISSN: 0022-0000. DOI: https://doi.org/10.1006/jcss.1997.1504. URL: https://www.sciencedirect.com/science/article/pii/S002200009791504X.

[44] McCallum, Andrew and Nigam, Kamal. "A comparison of event models for naive bayes text classification". In: *AAAI Conference on Artificial Intelligence*. 1998. URL: https://api.semanticscholar.org/CorpusID:7311285.

[45] 1.10. Decision Trees. URL: https://scikit-learn.org/stable/modules/tree.html#decision-trees.

[46] Breiman, L. "Arcing classifiers". In: ANNALS OF STATISTICS 26 (1998), pp. 801–823.

[47] Zhang, Harry. "The Optimality of Naive Bayes". In: *The Florida AI Research Society*. 2004. URL: https://api.semanticscholar.org/CorpusID:8891634.

[48] Friedman, Jerome H. "Greedy function approximation: A gradient boosting machine." In: The Annals of Statistics 29.5 (2001), pp. 1189–1232. DOI: 10.1214/aos/1013203451. URL: https://doi.org/10.1214/aos/1013203451.

[49] Goldberger, Jacob et al. "Neighbourhood Components Analysis". In: *Neural Information Processing Systems*. 2004. URL: https://api.semanticscholar.org/CorpusID:8616518.

[50] Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome. The elements of statistical learning. Jan. 2009. DOI: 10.1007/978-0-387-84858-7. URL: https://doi.org/10.1007/978-0-387-84858-7.

[51] Fan, Rong-En et al. LIBLINEAR: a library for large linear classification. Tech. rep. Aug. 2008, pp. 1871–1874. URL: https://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf.

[52] Chang, Chih-Chung and Lin, Chih-Jen. LIBSVM: a library for support vector machines. Tech. rep. 2001. URL: https://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf.

[53] Popescu, Marius-Constantin et al. "Multilayer perceptron and neural networks". In: WSEAS Transactions on Circuits and Systems 8 (July 2009).

[54] Rennie, Jason D. M. et al. "Tackling the poor assumptions of naive bayes text classifiers". In: *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*. ICML'03. Washington, DC, USA: AAAI Press, 2003, pp. 616–623. ISBN: 1577351894.

[55] Crammer, Koby et al. Online Passive-Aggressive Algorithms. Tech. rep. Mar. 2006, pp. 551–585. URL: https://jmlr.csail.mit.edu/papers/volume7/crammer06a/crammer06a.pdf.

[56] Breiman, Leo. “Random Forests”. In: Machine Learning 45.1 (Jan. 2001), pp. 5–32. DOI: 10.1023/a:1010933404324. URL: https://doi.org/10.1023/a:1010933404324.

[57] LeCun, Yann et al. Efficient BackProp. Springer, 1998. URL: https://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf.

[58] Ho, Tin Kam and Basu, M. “Complexity measures of supervised classification problems”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 24.3 (2002), pp. 289–300. DOI: 10.1109/34.990132.

[59] Arshad, Muhammad A, Shahriar, Sakib, and Anjum, Khizar. The Power Of Simplicity: Why Simple Linear Models Outperform Complex Machine Learning Techniques – Case Of Breast Cancer Diagnosis. June 2023. DOI: 10.48550/arXiv.2306.02449.

[60] Strang, Benjamin et al. “Don’t Rule Out Simple Models Prematurely: A Large Scale Benchmark Comparing Linear and Non-linear Classifiers in OpenML”. In: *International Symposium on Intelligent Data Analysis*. 2018. URL: https://api.semanticscholar.org/CorpusID:52959702.

	Dataset	Standard Accuracy	Modified Accuracy	Difference
0	splice	0.967085	0.952978	0.014107
1	breast-w	0.964286	0.971429	-0.007143
2	first-order-theorem-proving	0.548203	0.546569	0.001634
3	madelon	0.834615	0.821154	0.013462
4	jm1	0.807074	0.806615	0.000459
5	credit-approval	0.833333	0.840580	-0.007246
6	optdigits	0.984875	0.981317	0.003559
7	analcatdata_authorship	0.982249	0.976331	0.005917
8	qsar-biodeg	0.886256	0.829384	0.056872
9	letter	0.929000	0.939500	-0.010500
10	pc3	0.888179	0.878594	0.009585
11	wdbc	0.973684	0.956140	0.017544
12	vehicle	0.758824	0.858824	-0.100000
13	banknote-authentication	1.000000	0.992727	0.007273
14	Internet-Advertisements	0.966463	0.969512	-0.003049
15	ilpd	0.743590	0.743590	0.000000
16	dresses-sales	0.550000	0.560000	-0.010000
17	mfeat-zernike	0.822500	0.835000	-0.012500
18	har	0.975243	0.987864	-0.012621
19	kc1	0.848341	0.850711	-0.002370
20	GesturePhaseSegmentationProcessed	0.603038	0.659747	-0.056709
21	mfeat-factors	0.967500	0.942500	0.025000
22	churn	0.962000	0.958000	0.004000
23	ozone-level-8hr	0.921105	0.923077	-0.001972
24	wilt	0.983471	0.983471	0.000000
25	cylinder-bands	0.712963	0.740741	-0.027778
26	pendigits	0.989541	0.992724	-0.003183
27	credit-g	0.770000	0.750000	0.020000
28	spambase	0.943540	0.943540	0.000000
29	mfeat-pixel	0.957500	0.970000	-0.012500
30	kc2	0.857143	0.752381	0.104762
31	steel-plates-fault	0.763496	0.791774	-0.028278
32	segment	0.893939	0.900433	-0.006494
33	cmc	0.583051	0.562712	0.020339
34	adult	0.872761	0.867847	0.004914
35	diabetes	0.746753	0.727273	0.019481
36	eucalyptus	0.635135	0.608108	0.027027
37	pc1	0.918919	0.923423	-0.004505
38	blood-transfusion-service-center	0.806667	0.806667	0.000000
39	bank-marketing	0.895831	0.901139	-0.005308
40	mfeat-karhunen	0.965000	0.965000	0.000000
41	pc4	0.856164	0.893836	-0.037671
42	mfeat-morphological	0.747500	0.742500	0.005000

Table 7: Accuracy scores of the modified and standard system on each dataset. Bolded differences are significant.