UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CHRISTIAN SCHMITZ

# Meta-Learning for Classifier Ensemble Optimization

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof[a]. Dr[a]. Renata Galante

Porto Alegre
July 2024

**CIP — CATALOGING-IN-PUBLICATION**

Schmitz, Christian

Meta-Learning for Classifier Ensemble Optimization / Christian Schmitz. – Porto Alegre: PPGC da UFRGS, 2024.

67 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2024. Advisor: Renata Galante.

1. Meta-learning. 2. Ensembles. 3. Classification Tasks. I. Galante, Renata. II. Título.

*“It’s time to try defying gravity”*

— WICKED

## ACKNOWLEDGEMENTS

First, I want to express my gratitude to my beloved girlfriend, Victoria Elizabetha Alves, who is always by my side and encourages me to pursue my dreams. Special thanks to my parents, Sandra Winter Schmitz and Francisco Xavier Schmitz, and my sister, Julia Winter Schmitz, for their unwavering support throughout my academic journey and their understanding in situations where I had to sacrifice family time to focus on my studies. An honorable mention to my dog, Snow, who stood by my side in the office during the countless hours required to finish this thesis.

I also want to thank the professors at the Instituto de Informática from the Universidade Federal do Rio Grande do Sul who, even in the face of the greatest difficulties that we know exist in the academic field in Brazil, remain strong in teaching and researching with excellence. I want to express my gratitude to Professor Renata de Matos Galante for supervising this work. I also want to thank Julius Voggesberger from Universität Stuttgart for his patience and technical guidance, as well as Professor Edimar Manica from Instituto Federal do Rio Grande do Sul - Campus Ibirubá for his constant feedback and advice.

## ABSTRACT

As machine learning becomes more popular, it is natural for non-experts to desire to leverage machine learning for their tasks. However, selecting an algorithm and fine-tuning it to work well on a given task is complex and requires technical knowledge, which they usually lack. This issue is even more evident when ensembles are used, as the number of algorithms to choose from and hyperparameters to tune grows significantly. Ensembles are particularly useful in complex tasks that involve challenges such as class imbalance or high dimensionality, which are often encountered in domain-specific tasks. Thus, developing mechanisms that help the non-technical user choose and tune an ensemble model to fit a task is highly relevant in the area of machine learning. In this thesis, a novel framework is presented called Meta-CLEO, which uses meta-learning to create ensembles for new tasks by relating them to previously learned ones, thus leveraging classifier ensembles that worked well on similar tasks in the past. Ensemble-specific diversity metrics are also used to provide increased generalization. Experiments with 74 tasks evaluated different ensemble ranking functions based on ensemble performance and diversity metrics and compared Meta-CLEO's results with two baselines, Random Forest and AdaBoost. Results show that Meta-CLEO is equivalent to or outperforms the baselines in more than 75% of the evaluated tasks.



**Keywords:** Meta-learning. Ensembles. Classification Tasks.

**Meta-aprendizado para a otimização de ensembles de classificação**

## RESUMO

À medida que o aprendizado de máquina se torna mais popular, é natural que profissionais de outras áreas e que não são especialistas queiram aproveitá-lo em suas tarefas. No entanto, selecionar um algoritmo e ajustá-lo para que funcione bem em uma determinada tarefa é complexo e requer conhecimento técnico em aprendizado de máquina, que os profissionais de outros domínios em geral não possuem. Esse problema fica ainda mais evidente quando são usados *ensembles*, pois o número de algoritmos a serem escolhidos e de hiperparâmetros a serem ajustados aumenta significativamente. Os *ensembles* são particularmente úteis em tarefas complexas que envolvem desafios como desequilíbrio de classe ou alta dimensionalidade, que são frequentemente encontrados em tarefas específicas de domínio. Assim, o desenvolvimento de mecanismos que ajudem o usuário não técnico a escolher e ajustar um modelo de *ensembles* para se resolver uma tarefa é de grande relevância na área de aprendizado de máquina. Nesta tese, é apresentado um novo *framework* chamado Meta-CLEO, que usa o meta-aprendizado para criar *ensembles* para novas tarefas relacionando-as com tarefas aprendidas anteriormente, aproveitando *ensembles* de classificadores que funcionaram bem em tarefas semelhantes no passado. Métricas de diversidade específicas de *ensembles* também são usadas para proporcionar maior generalização. Os experimentos realizados com 74 tarefas avaliaram diferentes algoritmos de ranqueamento de *ensembles* com base no desempenho do *ensemble* e nas métricas de diversidade e compararam os resultados do Meta-CLEO com dois *baselines*, *Random Forest* e *AdaBoost*. Os resultados mostram que o Meta-CLEO é equivalente ou tem desempenho superior aos *baselines* em mais de 75% das tarefas avaliadas.

**Palavras-chave:** Meta-aprendizado, Ensembles, Tarefas de Classificação.

## LIST OF FIGURES

## LIST OF TABLES

**CONTENTS**

# 1 INTRODUCTION

Machine learning (ML) has never been so popular. With its increasing adoption and the evolution of hardware resources and cloud computing, more and more research areas and industries are expanding their technology investments to solve their domain challenges with ML. Use-cases with ML solutions are rapidly growing within agriculture (Ayoub Shaikh; RASOOL; Rasheed Lone, 2022; SHARMA et al., 2020), industry 4.0 (LASI et al., 2014; DALENOGARE et al., 2018), smart cities (ALANNE; SIERLA, 2022) and civil engineering (KHAMBRA; SHUKLA, 2023; AVCI et al., 2021), to name a few.

Many tasks can be solved through ML, e.g., regression, classification, and machine translation. One of the most relevant and the focus of this thesis is classification, i.e., the prediction of which group a given instance belongs to. Classification tasks are solved by classifiers, whose main challenge is identifying what makes samples different from one category to another and using the acquired knowledge to predict the class of an unknown instance. Some commonly used classifiers are neural networks (WIDROW; LEHR, 1990), decision trees (QUINLAN, 1986), and Nearest-Neighbors (COVER; HART, 1967).

The use of a single, isolated classifier may be limited (DIETTERICH, 2000), especially in complex scenarios (HIRSCH; REIMANN; MITSCHANG, 2019), where the amount of data is either huge/scarce (POLIKAR, 2006) and/or imbalanced (GALAR et al., 2011; HE; GARCIA, 2009; SAGI; ROKACH, 2018). By combining classifiers that complement each other in an ensemble, it is possible to increase the predictive accuracy and robustness of ML solutions (TUMER; GHOSH, 1996; ROKACH, 2010).

Ensembles achieve state-of-the-art results (SHIFAZ et al., 2020), and their success depends on how the base classifiers that compose them perform and the diversity between them (HANSEN; SALAMON, 1990). For base classifiers to be diverse, they must make correct and incorrect predictions on different data instances. Some common methods to achieve diversity are to train the base classifiers on different data samples or to use heterogeneous sets of base classifiers (SAGI; ROKACH, 2018), i.e., creating a heterogeneous ensemble. In addition to the base classifiers, the ensemble also requires a decision fusion mechanism that generates the ensemble's final prediction based on the prediction of the base classifiers. The decision fusion mechanism may be as simple as counting the base classifier predictions for each label and returning the most voted one. Or, it may be another ML algorithm that requires further training and tuning. There is no optimal fusion

mechanism for all tasks, and choosing the right one is critical to obtaining the best results (WILHELM et al., 2023).

Developing a classifier with high predictive performance to solve a given task is a challenging process. An algorithm must be picked, and its hyper-parameters must be tuned. For classifier ensembles, the challenge is even more complex, as multiple classification algorithms must be picked, each with its own hyper-parameters, and a decision fusion mechanism must be chosen on top, which may additionally need hyperparameter tuning. Machine learning experts rely on their expertise and past experience with various tasks to tackle this challenge and develop a solution for a new task. However, for novice analysts who still lack ML experience, creating a solution that solves the task with high accuracy can become a challenging process.

An approach that has been successfully used to make ML more accessible to novice analysts is meta-learning. Meta-learning, also known as learning to learn, is a subfield of ML that leverages metadata gathered from previously solved tasks and their ML solutions to learn new tasks more efficiently and accurately (VANSCHOREN, 2019). With meta-learning, it becomes possible to determine which algorithms performed well on similar tasks and use this information to produce a solution for a new task that was previously unknown. Even though meta-learning has been used in the context of classification tasks and ensemble models in the past (KHAN et al., 2020; SILVA et al., 2021), no literature was found that explores how heterogeneous classifier ensembles behave on a set of tasks so that a well-performing classifier ensemble can be created for a new task.

In addition, as the meta-learning-based ensemble solution for the new task is based on the ensembles that performed well on a similar task previously, it is possible that the best ensemble for the similar task is not the best one for the new task, i.e., it might not generalize well. To address this, ensemble ranking can be done, which involves selecting the best ensembles from a similar task according to a predefined ranking condition and pruning out the rest. Ranking of ensembles can be done through predictive accuracy but also diversity metrics. No research was found that explores the use of diversity when ranking classifier ensembles.

This thesis presents a novel framework that combines the past knowledge from meta-learning with the proven efficacy of ensembles, called Meta-CLEO (**Meta**-Learning for **Cl**assifier **E**nsemble **O**ptimization). By merging ensembles with meta-learning, it is possible to take advantage of key ensemble qualities, such as diversity, as well as ensemble components, such as the ensemble size, the base classifiers, and the decision fusion

method, as unique metadata for meta-learning. The contribution of this work includes:

- A novel way of combining meta-learning and heterogeneous ensembles to assist non-technical ML users in finding a well-performing ensemble solution for their tasks;
- The proposal and evaluation of a ranking system based on ensemble performance and diversity metrics that provide well-performing ensembles while reducing run time complexity; and
- The evaluation of Meta-CLEO against two widely known ensemble techniques commonly used in ML: Random Forest and AdaBoost;

Experiments with 74 different tasks with various data set sizes and data characteristics show that more than 70% of the tasks benefit from Meta-CLEO when compared to Random Forest and 94% when compared to AdaBoost.

The remainder of this thesis is structured as follows: Chapter 2 dives into the core concepts around meta-learning and ensembles while also discussing related work. In Chapter 3, a set of requirements for the solution are defined, and three possible approaches that led to Meta-CLEO are described. Chapter 4 presents Meta-CLEO, describing its offline and online stages, the blend of ensembles and meta-learning, and classifier ensembles ranking. The implementation of Meta-CLEO and the conducted experiments are explained in Chapter 5. Lastly, in Chapter 6, the conclusion and future work are delineated.

# 2 BACKGROUND AND RELATED WORK

In this chapter, the core concepts of the current work, as well as the related work, are presented. In Section 2.1, classifier ensembles and their components are detailed. Section 2.2 explains what ensemble diversity is and how to calculate it. Then, in Section 2.3, meta-learning is introduced, and the general architecture of a meta-learning application is explained. Streamlining the previous sections, Section 2.4 describes related work combining ensembles and meta-learning, while in Section 2.4.1, a comparison between related work is performed, and their distinguishing factors are highlighted. Finally, in Section 2.5, we streamline the concepts, the related work, and the target of the proposed framework.

## 2.1 Ensemble of Classifiers

Machine Learning aims to learn from data (MAHESH, 2020). The outcome of this learning process can be, however, diverse. One may want to predict a continuous value, e.g., a stock price, which is considered an ML regression task. Others may want to predict a group to which a data sample belongs, e.g., whether a stock is worth buying at a point in time, which is considered a classification task. As this thesis centers on classification tasks, we will focus on ML algorithms that solve them, known as classifiers.

A classifier is a function $D : \boldsymbol{x_j} \rightarrow y_j$ that takes a data instance $\boldsymbol{x_j}$ from a labeled data set $Z = \{(\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_m}, y_m)\}$ and assigns a label $y_j \in \Omega$ from the set of *v* labels $\Omega = \{w_1, ..., w_v\}$ to it. The main challenge for supervised classifiers is to correctly draw the classification boundary in the feature space of a data set $Z_f$ that delimits the labels from each other.

To better cope with complex scenarios such as those with many classes or noisy data, leveraging multiple classifiers is sometimes useful. Combining multiple classifiers into one to generate a prediction is often called multiple classifier systems (MCS) (HO; HULL; SRIHARI, 1994) or ensembles of classifiers. To build an ensemble of classifiers $e_p$, in addition to the base classifiers $\boldsymbol{D} = \{D_1, ..., D_L\}$, a decision fusion mechanism must be selected. The decision fusion mechanism is a function $C(x) = F[D_1(x), ..., D_L(x)]$ that combines the outputs of the individual classifiers and defines the final ensemble prediction. The choice of the best fusion mechanism depends on the task at hand. Wilhelm et al. (2023) propose three categories of decision fusion, which are:

- **Utility-based Decision Fusion**, which directly applies to classifiers' predictions. One example of a utility-based decision fusion method is *Majority Voting*, which combines the predictions of the base classifiers by choosing the class label that most of the classifiers predict, i.e., the majority vote.

- **Evidence-based Decision Fusion** requires additional information about the classifiers to provide the final prediction. For example, in *Weighted Voting*, the output of the base classifiers is weighted depending on previous knowledge of their performance to define the final ensemble prediction.

- **Trainable Decision Fusion**, as the name indicates, requires the training of an algorithm on top of the base classifiers' predictions before being able to predict the final output of the ensemble. *Stacked Generalization* is a trainable decision fusion mechanism in which an ML model is trained to learn how to best combine the outputs of the base classifiers of the given ensemble (WOLPERT, 1992).

Thus, an ensemble of classifiers $e_p$ trained on a data set $Z_f$ can be seen as a tuple $e_p = (D_p, C_p)$ composed of the set of trained base classifiers $D_p$ and the decision fusion mechanism $C_p$.

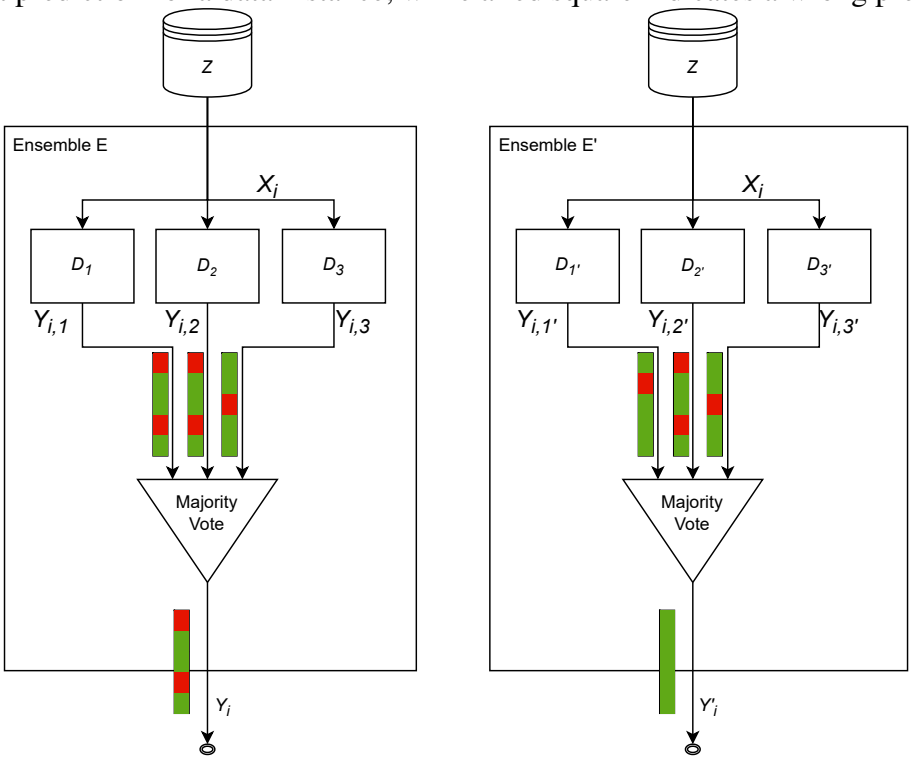
## 2.2 Ensemble Diversity

Assembling the building blocks of an ensemble of classifiers does not necessarily generate accurate outcomes. When creating accurate ensembles of classifiers, the base classifiers must perform well *and* be diverse (HANSEN; SALAMON, 1990). Ensembles are diverse when the ensemble's base classifiers succeed and fail on different data instances.

Figure 2.1 illustrates the impact of diverse base classifiers and the influence of the decision fusion mechanism on ensemble prediction. When classifiers are diverse, they tend to make complementary errors, which allows the ensemble's decision fusion mechanism to make the correct final prediction by properly fusing the base classifiers predictions. For instance, let's consider the ensemble $E$ from Figure 2.1. When classifiers $D_1$ and $D_2$ predict $Y_{i,1} = Y_{i,2} = \{incorrect, correct, correct, incorrect, correct\}$, the final output of the ensemble after a majority voting fusion is $Y_i = \{incorrect, correct, correct, incorrect, correct\}$. This output is heavily influenced by the results of the first

two classifiers. In contrast, if the base classifiers in an ensemble are diverse and make uncorrelated errors, such as $E'$, the decision fusion mechanism can better deal with incorrect predictions and try to make the correct prediction prevail.

Figure 2.1: Example of the impact diversity has on ensembles. A green square indicates a correct prediction for a data instance, while a red square indicates a wrong prediction.




Source: The Authors

The generation of diversity can be either explicit or implicit (BROWN et al., 2005). The first occurs when diversity information is used to optimize diversity directly during ensemble creation. The latter occurs when diversity is expected to emerge without using any explicit measurement to ensure diversity is created, e.g., by influencing the training process. This is often done using different training data for each classifier, e.g., through bootstrapping or using heterogeneous base classifiers.

Explicit diversity can be achieved by optimizing the ensemble through a diversity metric (VOGGESBERGER; REIMANN; MITSCHANG, 2023). Diversity metrics measure the diversity within a set of classifiers. They are split into pairwise and non-pairwise metrics, depending on whether they are calculated between each pair of classifiers and then aggregated or between the entire set of classifiers at the same time. Examples of standard pairwise metrics are the *disagreement* metric, which measures the probability that the two classifiers diverge on their decisions, and the *double fault* metric, which measures the probability of both classifiers predicting wrong outputs simultaneously (KUNCHEVA,

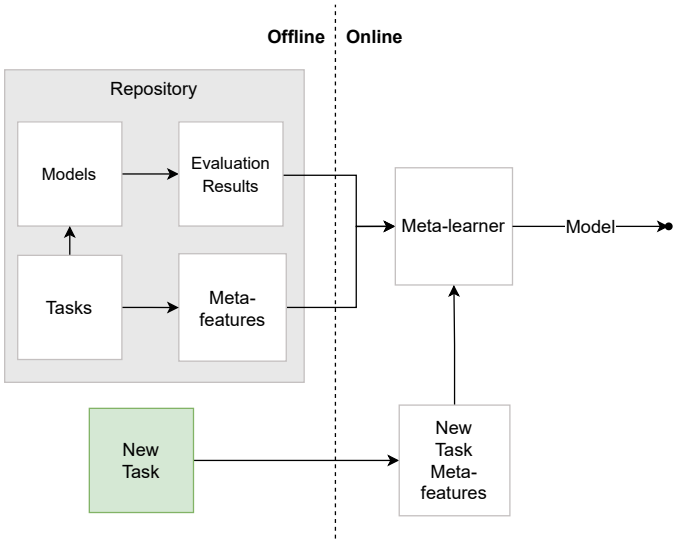
2004). On the other hand, non-pairwise metrics such as *entropy* (CUNNINGHAM; CARNEY, 2000) and the *Kohavi-Wolpert variance* (KOHAVI; WOLPERT, 1996) rely on the entire ensemble to calculate their measurements.

There are many approaches to achieving diversity when creating an ensemble. One technique that implicitly achieves diversity by manipulating training data is bagging (BREIMAN, 1996). In bagging, the base classifiers are trained on bootstrap samples, which are drawn randomly and with replacements from the training set for each classifier. This makes it harder for classifiers to overfit the training data set. Last, the base classifiers' outputs are combined through majority voting to form the overall ensemble prediction. Random Forests (BREIMAN, 2001) are a sub-type of bagging in which the base classifiers are trees. The training data can be sampled within random forests on the instance level, feature level, or both. For example, the Random Subspace Method (HO, 1998) performs feature level sampling by randomly picking identically distributed feature sets that will be used to train each tree from the forest.

Another well-known algorithm that targets the creation of diverse ensembles is boosting (FREUND; SCHAPIRE; ABE, 1999). In boosting, each base classifier is trained in a way that tries to correct the errors made by previous classifiers. The most famous boosting algorithm is AdaBoost (FREUND; SCHAPIRE, 1997), which explicitly optimizes diversity by re-weighting the training data distribution for each classifier so that instances that were classified wrongly by a classifier will be weighted more when drawing the training set for the following classifiers. In addition, AdaBoost also weights the votes from its base classifiers, giving more influence to those with more correct predictions on more complex instances. XGBoost (CHEN; GUESTRIN, 2016) and CatBoost (PROKHORENKOVA et al., 2018) are two ensemble algorithms based on boosting that achieve state-of-the-art performance (SHWARTZ-ZIV; ARMON, 2022). The former doesn't reweigh data distribution but instead uses Gradient Boosting (FRIEDMAN, 2001) to optimize the next classifier model based on the residual errors of its predecessor. The latter also uses gradient boosting, but it is designed to perform better, especially with categorical and textual data.

In Meta-CLEO, we aim to leverage ensemble diversity when suggesting an ensemble for a new task. With that in mind, we must have ways of measuring diversity within ensembles. Therefore, tackling diversity explicitly becomes essential.

Figure 2.2: General architecture of a meta-learning application. Adapted from Smith-Miles (2009).




## 2.3 Meta-learning

Meta-learning is a machine-learning subfield inspired by how humans learn from past experiences to solve new problems (THRUN; PRATT, 1998). In meta-learning for classification, metadata is extracted from previously solved tasks to allow them to be related to new ones based on similarity (MONTEIRO et al., 2021). The algorithms that perform best in solving these similar tasks are then used to solve the new ones, accelerating the process. Meta-learning is a powerful technique that can help improve efficiency and effectiveness when solving new tasks (VANSCHOREN, 2019).

The typical design of a meta-learning application can be seen in Figure 2.2. It is composed primarily of a repository and a meta-learner. The repository stores the metadata from the previously learned tasks, the ML models used, and their evaluation results. As the repository creation can be time-consuming, it occurs *offline* in what is sometimes called the *meta-learning training stage*.

The repository is then used to train the meta-learner, the algorithm responsible for learning the relationship between the meta-features and the model's evaluation results for a task. The meta-learner can be an ML algorithm such as kNN or a rule-based system that, given a new task's meta-features, returns a model based on the repository data. The meta-learner prediction happens *online*, i.e., during runtime, and is sometimes called the *meta-learning generalization stage*.

The repository data plays a crucial role in the meta-learning application. It serves as the training data for the meta-learner and comprises many attributes that characterize

the task and its solution, including metadata about 1) the trained model, such as its performance on test data, the algorithm, and hyperparameters used during training, and 2) the task, such as the data set's number of instances, number of classes, and the ratio between them.

The metadata extracted from the tasks are called meta-features. Many meta-features have been proposed in addition to the most simple ones mentioned above, often called *general* or *simple* meta-features (REIF et al., 2014). In Rivolli et al. (2019), the authors created a taxonomy for grouping them based on similarity. The most prominent meta-feature groups are:

- *General* meta-features, which are fast and cheap to compute and well-known for data set characterization in ML literature. This category includes the number of the data set's binary and categorical features, for example (MICHIE et al., 1995; ENGELS; THEUSINGER, 1998).

- *Statistical* meta-features indicate statistical aspects of the data, such as the mean, median, and variance (ALI; SMITH, 2006) of each feature.

- *Information-theoretic* meta-features come from the information theory field and are usually entropy-related, e.g., the features or target entropy (MICHIE et al., 1995) of the data set.

- *Landmarking* meta-features apply fast classification algorithms to the data set and use their performance results as meta-features. For example, the performance of applying kNN with $k = 1$ to the data set (BENSUSAN; GIRAUD-CARRIER, 2000).

- *Model-based* meta-features, in contrast to landmarking, apply classification algorithms to the data set and collect its properties to be used as meta-features. Decision Trees are commonly used for this purpose and generate meta-features such as the number of leaf or non-leaf nodes in the induced model (PAVEL; SOARES, 2002).

## 2.4 Related Work

In this section, work related to Meta-CLEO is presented. As mixing the previously presented topics, ensembles and meta-learning, has not yet been deeply explored, there is little literature about it. Therefore, we will highlight the identified tools that integrate

these two areas from their own perspectives. After, in Subsection 2.4.1, a comparison between the tools is provided.

In the context of ensembles and meta-learning, Meta-DES (CRUZ et al., 2015) proposes a framework that uses meta-learning to perform Dynamic Ensemble Selection (DES) (KO; SABOURIN; JR, 2008), i.e., select, for each new test data instance, the classifiers from an ensemble that will be predicting on it. Although less common, in Meta-DES, the repository is built with a single task by extracting the meta-features from the current task data set during the training phase.

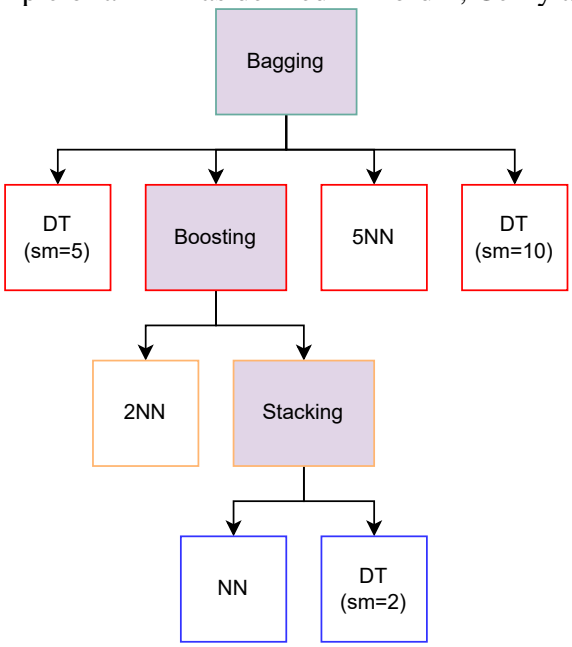
So, for a given task, the framework uses the training data set to train an ensemble *and* the meta-learner, which labels the ensemble's base classifiers as competent or not for a given instance. Then, in the test phase, given an instance from the test data set, the meta-learner elects the competent models, and their outputs are fused through majority voting to form the final ensemble prediction.

Some approaches do not only create classifiers using meta-learning but also continue to optimize them using automated machine learning (AutoML) (FEURER; HUTTER, 2019). AutoML tools select and optimize an algorithm for a given task provided by a user, with minimal or no human interaction, by automating the time and cost-intensive optimization process. In general, AutoML tools solve the problem of combined algorithm selection and hyperparameter optimization (CASH) (THORNTON et al., 2013) by selecting an algorithm and optimizing its hyperparameters simultaneously for a task, trying out many algorithm configurations exhaustively, and providing the most accurate within a given time or resource budget.

Auto-Sklearn (FEURER et al., 2015) is one of the first AutoML frameworks created, which utilizes meta-learning and ensembles. In Auto-Sklearn, the repository contains metadata from different tasks' data sets and their optimized solutions. Given a new task, Auto-Sklearn ranks repository entries similar to the new task's data set metadata and selects the best algorithm configurations from the top-$k$ similar tasks. Models created from these algorithm configurations are then optimized through AutoML for the new task, and ensemble selection (CARUANA et al., 2004) is performed to select the subset of models that will take part in the final ensemble.

Similarly, Kordík, Černỳ and Frỳda (2018) utilize meta-learning and AutoML to construct ensembles. The repository is created by optimizing ensembles for the repository tasks. The optimization occurs by evolving Meta-learning Templates (MATs) through genetic programming. MAT is a hierarchical structure representing the architecture of

Figure 2.3: Example of a MAT as defined in Kordík, Černỳ and Frỳda (2018).




an ensemble of classifiers defined by the authors. Figure 2.3 shows one example of a MAT. For a new task, an initial population of MATs is drawn from the repository based on similar metadata and evolved to create a new, optimized ensemble.

ERM-ML (Ensemble Recommendation Method - Using Meta-Learning) and ERM-3ML (Ensemble Recommendation Method - Using 3 Steps Meta-Learning) (SILVA et al., 2021) use meta-learning to provide automatic recommendations of ensemble structures, i.e., the type of the base classifier, ensemble size, and the decision fusion method. In ERM-ML, the ensemble structure is predicted entirely by a single meta-learner, while in ERM-3ML, it is sequentially predicted through three different meta-learners. With this sequential approach, ERM-3ML uses the prediction of the previous meta-learner as a meta-feature for the following meta-learners, enriching the context and thus providing better results.

### 2.4.1 Comparison

Table 2.1 summarizes how the related work integrates with the core concepts presented in Sections 2.1-2.3. The following aspects were considered: whether the tool 1) varies ensemble size, heterogeneous base classifiers, and fusion mechanisms while creating ensembles; 2) tackles diversity, an essential part of successful ensembles, explicitly; and 3) uses meta-learning to leverage similar tasks' ensemble solutions when creating a

classifier for a new task.

Table 2.1: Comparison between the related work. ● = fully supported, ◒ = partly supported, ○ = not supported.

| Approach | Ensemble Creation | Explicit Diversity | Previous Knowledge |
|---|---|---|---|
| Meta-DES (CRUZ et al., 2015) | ◒ | ○ | ◒ |
| Auto-Sklearn (FEURER et al., 2015) | ◒ | ○ | ● |
| MAT (KORDÍK; ČERNÝ; FRÝDA, 2018) | ● | ◒ | ● |
| ERM-3ML (SILVA et al., 2021) | ◒ | ○ | ● |

Although research exists in the area of DES with meta-learning (CRUZ et al., 2015; CRUZ; SABOURIN; CAVALCANTI, 2014; CRUZ; SABOURIN; CAVALCANTI, 2017; RIJN et al., 2015; MA et al., 2020), this approach is limited as it often requires a pre-defined, one-size-fits-all ensemble structure to be trained for each new data set, so no knowledge from previously learned tasks are leveraged to learn new tasks. Auto-Sklearn leverages knowledge from previous tasks but only uses meta-learning to optimize classifiers, not the ensemble, such as warm-starting the AutoML process. It also generates ensembles but does not vary the decision fusion mechanism or tackle diversity.

In MAT, despite handling ensembles with various base classifiers and decision fusion mechanisms, diversity is only explicitly tackled when the underlying MAT consists of a method that handles it that way, e.g., when the MAT is composed of a boosting algorithm. Finally, ERM-ML and ERM-3ML do not tackle diversity explicitly. In addition, they also create homogeneous ensembles, i.e., the algorithm used in the base classifiers is always the same.

## 2.5 Final Remarks

Classifier ensembles can provide enhanced results if compared to individual classifiers, especially when they are diverse. Additionally, leveraging meta-learning, i.e., the experience gained from solving previously learned tasks, allows the creation of better models more efficiently. So, by combining diverse ensembles of classifiers with meta-learning, it is possible to assemble a framework that helps non-technical users of machine learning to create ensemble models that accurately solve their tasks. No related work was found that further explores these concepts together. In the upcoming chapter, we will present three possible approaches for creating the target framework. We will compare their designs and select the most suitable one to serve as the foundation for Meta-CLEO.

# 3 PROPOSED APPROACHES

This thesis aims to combine diverse ensembles with meta-learning to ease the use of ML by non-technical users. To allow the creation and evaluation of feasible alternatives toward the defined goal, a set of requirements must be created. These requirements serve as boundary conditions for the approaches to be proposed. This chapter defines such requirements, proposes three approaches, and compares them.

The remainder of this chapter is organized as follows: first, in Section 3.1, we define the set of requirements the selected approach must meet. Then, Section 3.2 describes a general approach, which serves as the base architecture for the following approaches. In Sections 3.2-3.5, we propose three potential approaches and provide further details about them. To conclude, in Section 3.6, the approaches are compared concerning the requirements, highlighting the one that fits best and will be used throughout the rest of this work as the foundation of Meta-CLEO.

## 3.1 Problem Statement

To achieve the goal of combining diverse ensembles and meta-learning to facilitate non-technical users to leverage ML, a new framework is required. This framework must generate an ensemble solution for a new classification task by utilizing previous knowledge through meta-learning. In addition, it must allow ensemble creation with heterogeneous classifiers and decision fusion mechanisms and explicitly tackle diversity. To clarify those aspects and precisely define the expectations for the selected approach, we created a set of requirements that must be met:

(R1): *Classifier Performance* The first requirement is for the base classifiers that compound the ensemble to be accurate. For a base classifier to be accurate, it must perform better than random guessing.

(R2): *Classifier Diversity* The second requirement is for the base classifiers to be diverse. For base classifiers to be diverse, they must make correct and incorrect predictions on different data instances. As the generated ensembles should be diverse, there must be a way to measure diversity within the ensembles. Therefore, the developed framework must focus on explicit diversity.

(R3): *Ensemble Fusion* The decision fusion method is substantial in ensemble creation,

and thus, the framework has to support multiple decision fusion methods.

(R4): *Meta-learning* The fourth requirement is to leverage knowledge acquired in previously learned tasks. For this, a set of information from previous tasks and their solutions is required, composed of the following:

(a) Metadata describing the task. Also known as meta-features.

(b) The algorithm selected to solve the task, as well as its configuration and hyperparameters.

(c) The performance measure(s) for the selected algorithm applied to the respective task.

(R5): *Runtime Complexity* The goal of the framework is to provide well-performing solutions to non-technical users and not fully optimized ones. Hence, the framework must execute fast, i.e., the framework must not exhaust possibilities during runtime to suggest the best one.

(R6): *Ease of Use* To make the framework accessible and friendly for non-technical users, the framework must be easy to use. The user must be able to effortlessly define the task to be handed to the approach. In addition, the resulting suggested ensemble must allow easy adoption by the user, be it for final use or further optimization.

(R7): *Extensibility* The framework must be openly extensible. Extensibility can be met by adding new meta-features to the meta-learning repository, new ensemble configurations, and decision fusion methods without altering the core of the framework. This is essential as numerous classification algorithms, decision fusion mechanisms, and performance and diversity measures exist that could be leveraged by extending the initial approach.

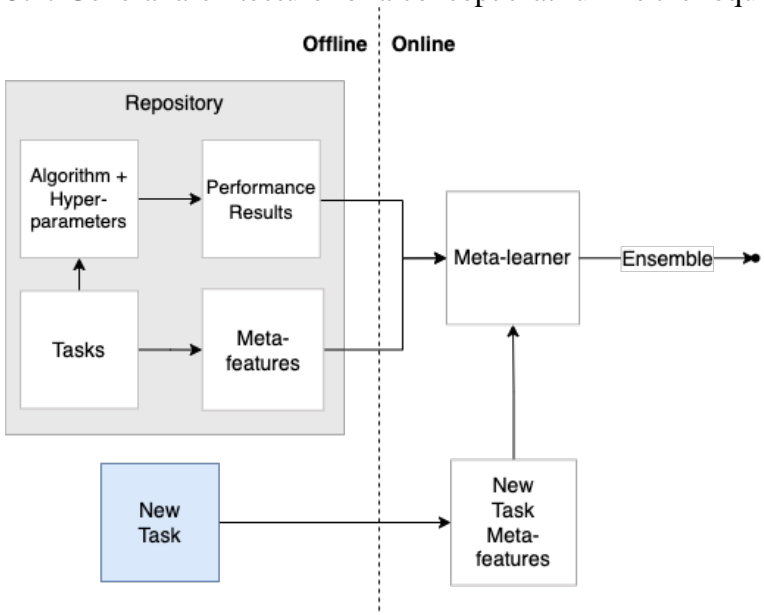
(R8): *Feasible Development Scope* The last requirement is for the framework to be developed within the scope of this Master Thesis.

## 3.2 General Approach

Based on the requirements and the typical architecture of meta-learning applications, it is possible to create a general framework architecture. Figure 3.1 depicts the general approach, which comprises of:

- A framework that takes a classification task as input, which includes an associated data set, and returns a classifier ensemble as output.
- A way of generating meta-features from the given task. The meta-features must be the same for the new task as the ones for the tasks available in the repository.
- A metadata repository containing, at least, the repository tasks, their meta-features, the algorithms and hyper-parameters used to learn them, and performance metrics.
- A meta-learner that is responsible for relating the meta-features from the tasks to the solution algorithm and its hyper-parameters and, given a new task, returning an ensemble trained on the data from the new task.

Figure 3.1: General architecture for a concept that fulfills the requirements.




Three extension approaches were developed, complementing the general architecture. The approaches expand the general architecture by specializing it in one or more of the components. In addition, the derivative approaches may include additional steps to achieve the desired results. The three approaches are described in the next Sections.

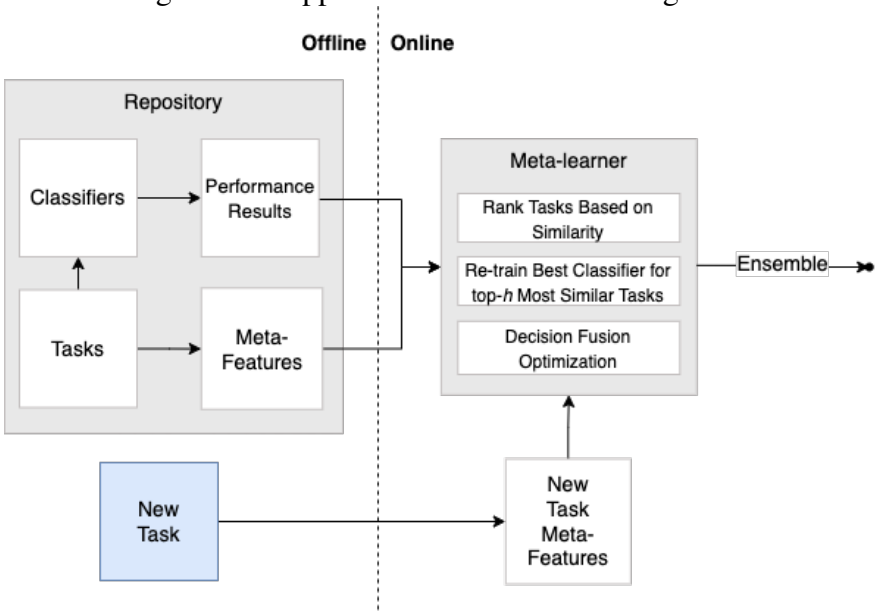
## 3.3 Approach 1 - Classifiers as Algorithms

This approach creates the repository by training and storing the classifier that performs best on each repository task. Hence, no ensembles are created in the offline stage. Instead, when a new task is received, the meta-learner creates a rank based on the similarity of the new task to the repository tasks. The top-$h$ classifiers from the rank are selected

to form the ensemble. Later, the decision fusion method is chosen by exhaustively trying all available alternatives and returning the one with the highest predictive performance in the new task. The meta-learner then assembles the selected classifiers and the decision fusion method and provides the trained ensemble as output.

Figure 3.2 illustrates the classifiers as algorithms approach. This approach is similar to what is done in Auto-Sklearn (FEURER et al., 2015). However, multiple decision fusion mechanisms are considered here.

Figure 3.2: Approach 1 - Classifiers as Algorithms




## 3.4 Approach 2 - Ensembles as Algorithms

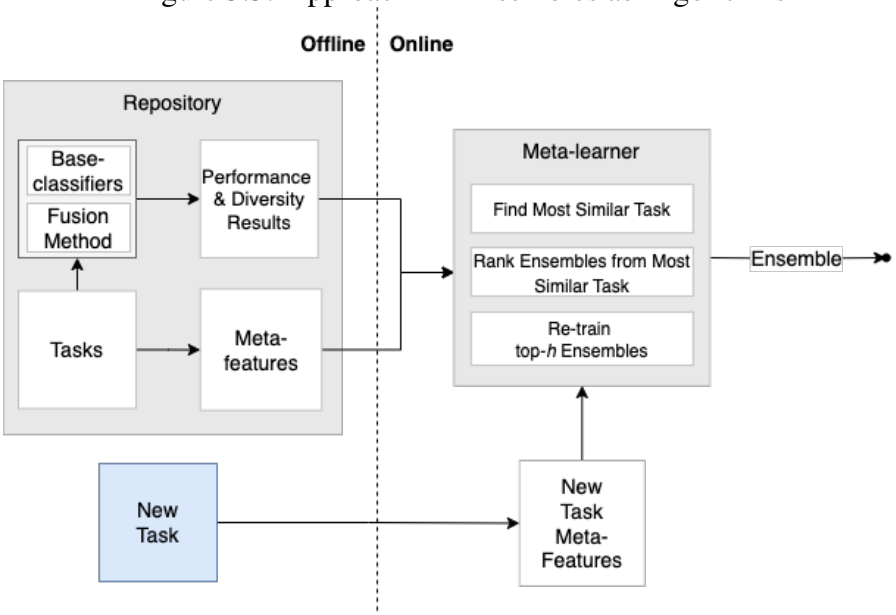
This approach creates the repository by training and storing all learned ensembles for each repository task. The ensembles are composed of heterogeneous base classifiers and a decision fusion method.

When a new task is received, the meta-learner identifies the most similar task and ranks its ensemble solutions based on performance **and** diversity measures. Then, top-*h* ensembles from the rank are trained in the train data set of the new task and evaluated in the respective test data set. The best-performing ensemble is selected as the output of the meta-learner.

Figure 3.3 illustrates the ensembles as algorithms approach. Explicit diversity is included during ensemble ranking, aiming at solutions that are good at their original task and that also generalize well. The balance between predictive performance and general-

ization can be tuned by adjusting the ranking function of the meta-learner.

Figure 3.3: Approach 2 - Ensembles as Algorithms

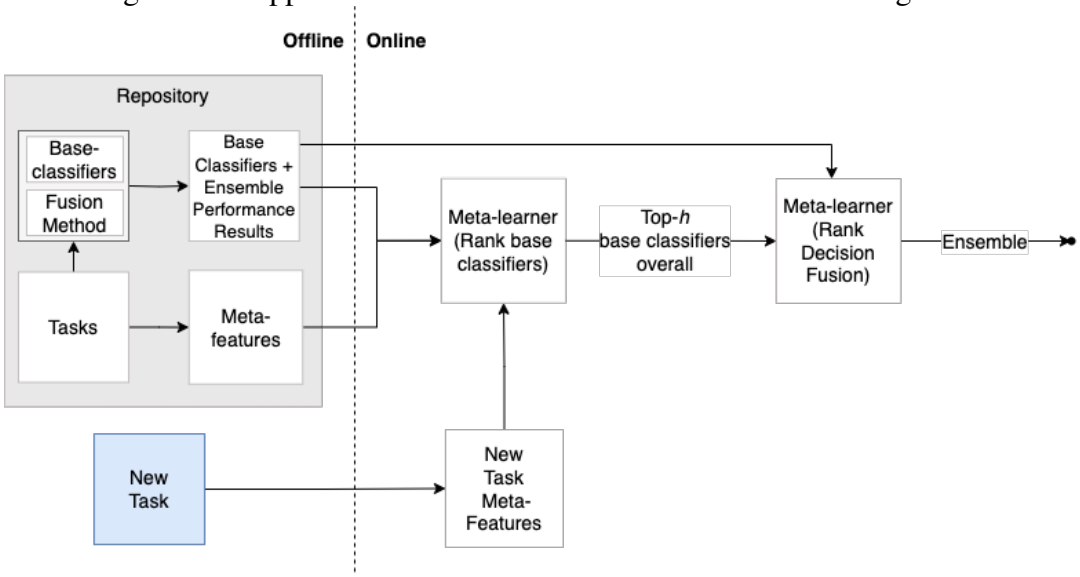



### 3.5 Approach 3 - Ensembles and Base-classifiers as Algorithms

This approach combines the first two, including two meta-learners. The repository is created by training and storing all learned ensembles for each repository task, as done in Approach 2. However, unlike Approach 2, every ensemble combination must be tested and stored, including all classifiers and decision fusion methods. This ensures that every combination of classifiers and decision fusion methods will have many evaluations stored in the repository, one for each task. In addition, metrics from the ensembles and their base classifiers are stored in the repository.

Figure 3.5 illustrates Approach 3. When given a new task, the first meta-learner ranks the individual base classifiers from the most similar task and selects the top-$h$ that will form a new ensemble. Then, the second meta-learner defines the decision fusion method by searching in the repository which method works best for the set of selected algorithms over all available repository solutions from all tasks. After selecting the decision fusion mechanism, the second meta-learner assembles the new ensemble, re-trains it with the training data from the new task, and returns it as the suggested ensemble.

Figure 3.4: Approach 3 - Ensembles and Base-classifiers as Algorithms




## 3.6 Comparison

Table 3.1 compares the three approaches concerning the requirements defined in Section 3.1. All approaches handle *R1 - Classifier Performance*. Approaches 1 and 3 tackle performance individually for each base classifier, while Approach 2 evaluates it on an ensemble level. Approaches 1 and 3 do not meet the *R2 - Classifier Diversity* requirement since the base classifier selection step happens independently for each base classifier and does not consider their composition. This occurs due to the exploding cost of considering all combinations of base classifiers and their diversity. Approach 2 fulfills the diversity requirement by including diversity measures in its ranking system, explicitly including diversity in the meta-learning process.

*R3 - Classifier Combination* is achieved in Approach 1 with the extra effort of extensively trying out different fusion mechanisms for the selected top-*h* base classifiers, which has a negative influence in *R5 - Complexity* as well. In approaches 2 and 3, many decision fusion mechanisms are already available in the repository solutions that are used to define the resulting ensemble, avoiding the need to try all decision fusion mechanisms in the new task before providing the final result.

All approaches leverage experience from previously learned tasks, required by *R4 - Meta-learning*. However, Approach 3 requires many more solutions in the repository, if compared to the other two approaches, so that the second meta-learner works as expected. This leads to a higher development scope, as generating enough repository data would take more computing hours of task learning. In terms of *R5 - Runtime Complexity*,

as already delineated, Approach 1 includes an optimization step that tries out all possible decision fusion mechanisms before deciding on a final one, increasing runtime complexity. Approaches 2 and 3 overcome this by restricting the alternatives that will be tried or picking the one with the highest predictive performance in previous tasks.

All approaches are similar w.r.t. *R6 - Ease of Use*, as the input and output are kept similar and allow further optimization. Regarding *R7 - Extensibility*, approaches 2 and 3 allow new classifiers and decision fusion mechanisms to be added by simply including solutions with the new techniques for the repository tasks. However, in Approach 3, the amount of samples to be added to the repository is significantly larger, as there must be a new solution with, e.g., the new decision fusion method for each combination of base classifiers. Approach 1 can be extended similarly for its base classifiers. The extensibility of the decision fusion mechanism from Approach 1 would require adding the new method to the list of methods to be tried out for each new task. All approaches allow meta-feature extensibility by enriching the repository data and parameterizing the meta-learner(s). *R8 - Feasible Development Scope* is not achievable for Approach 3 as the combinations of base classifiers and decision fusion methods quickly explode when every combination must be available in the repository. In contrast, approaches 1 and 2 have a feasible development scope for a Master Thesis as their repositories are more flexible regarding the number of algorithm configurations and thus require fewer computing hours to build.

Table 3.1: Comparison of the proposed approaches w.r.t. requirements presented in Section 3.1. ● = fully supported, ◒ = partly supported, ○ = not supported.

| Requirement | Approach 1 | Approach 2 | Approach 3 |
|---|---|---|---|
| R1 - Classifier Performance | ● | ● | ● |
| R2 - Classifier Diversity | ○ | ● | ○ |
| R3 - Classifier Combination | ● | ● | ● |
| R4 - Meta-Learning | ● | ● | ● |
| R5 - Complexity | ○ | ● | ● |
| R6 - Ease of Use | ● | ● | ● |
| R7 - Extensibility | ● | ● | ○ |
| R8 - Development Scope | ● | ● | ○ |

After considering the limitations of approaches 1 and 3, approach number 2 was selected as the foundation of Meta-CLEO. This approach not only meets all the requirements outlined in Section 3.1 but also allows for a new perspective on meta-learning in algorithm selection, specifically by focusing on heterogeneous ensembles of classifiers.

# 4 META-LEARNING FOR CLASSIFIER ENSEMBLE OPTIMIZATION

In this Chapter, Meta-CLEO is presented. Meta-CLEO is a meta-learning-based framework that creates ensemble solutions for classification tasks. It uses metadata collected across different tasks and their ML solutions, composed of heterogeneous ensembles with a wide range of base classifiers, ensemble sizes, and fusion mechanisms. In Meta-CLEO, diversity is achieved explicitly by including diversity metrics as metadata in the repository and leveraging them in the meta-learner.

## 4.1 Overview

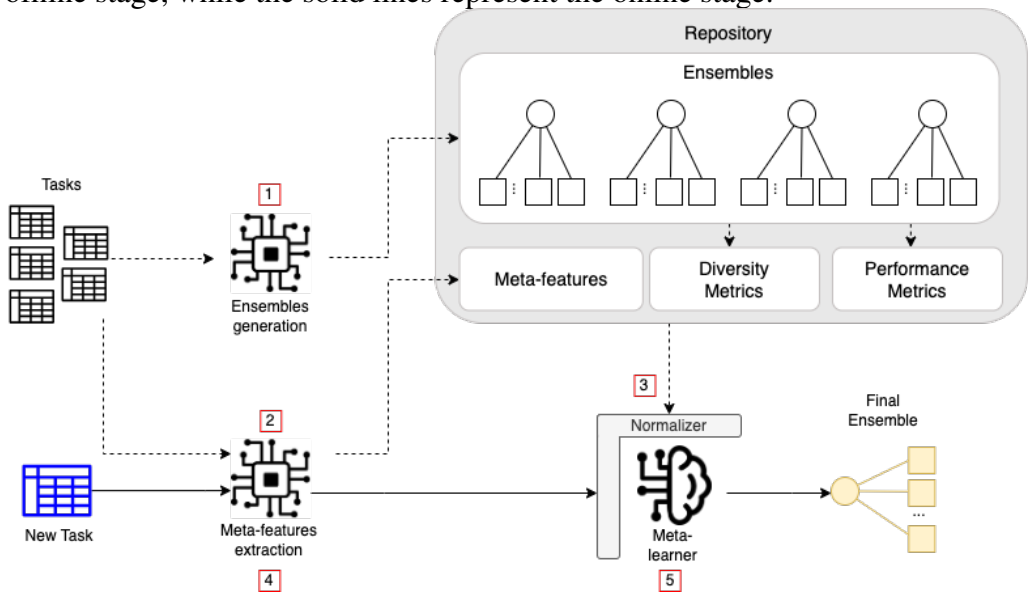
Figure 4.1 provides an overview from the workflow of Meta-CLEO. The numbers represent the order in which the flow is executed. Steps 1-3 are executed in the offline phase, while steps 4 and 5 occur in the online phase (cf. Section 2.3). The new task, in blue, is the classification task received as input. A task contains the train and test data sets, as well as a task identifier. In steps 1 and 2, the ensembles for the repository tasks are generated, their metadata is extracted, and their meta-features are calculated. Then, in step 3, the repository data is used to train the meta-learner. During the online phase, the user inputs a new task, which has its meta-features extracted in step 4, normalized, and handed to the meta-learner in step 5. The meta-learner identifies the most similar task, ranks its classifier ensembles, re-trains the top-$h$ ensembles on the training data of the new task, and returns the most accurate one.

To allow a better understanding of how Meta-CLEO works internally, the following sections describe the two stages of Meta-CLEO. The first stage, described in Section 4.2, is the offline stage, which includes creating the repository and training the meta-learner. The second is the online stage, which is described in Section 4.3 and involves receiving a new task, extracting its meta-features, and using the meta-learner to create a classifier ensemble solution.

## 4.2 Offline Stage

In the offline stage, the repository is created, and the meta-learner is trained. The repository is required to train the meta-learner, which, during the online stage, is respon-

Figure 4.1: High-level architecture overview of Meta-CLEO. The dashed lines represent the offline stage, while the solid lines represent the online stage.




sible for providing the classifier ensemble solution.

The first step towards creating the meta-learner is to build the repository. The repository is a set $M = \{m_{1,1}, m_{1,2}, ..., m_{p,s}\}$ containing the metadata $m_{t,r}$ from a task $t$ and a classifier ensemble solution $r$. A repository entry $m_{t,r}$ is defined as follows:

$$m_{t,r} = \{tm_t, em_r\} \tag{4.1}$$

$$tm_t = \{id_t, mtf_t\} \tag{4.2}$$

$$em_r = \{ec_r, perf_r, div_r\} \tag{4.3}$$

As stated in Equation 4.1, metadata $m_{t,r}$ contains the metadata $tm_t$ of a task $t$ and ensemble solution metadata $em_r$ from an ensemble $r$. The subset $tm_t$ (Equation 4.2) is the data set metadata from a task $t$ and is composed of the data set identifier $id_t$ and the meta-features $mtf_t$ that characterize the data set of the task.

The meta-features that are most suitable for Meta-CLEO are the ones that can be computed quickly. As explained in Section 2.3, model-oriented meta-features, like landmarking, need to train an ML model for each data set to extract metadata from it. This process can take a long time and slow down the framework during the online stage when it needs to quickly extract meta-features for the data set of a new task, which is essential for *R5 - Runtime Complexity*. Therefore, the simple, statistical, and information theory meta-feature groups best fit.

A single meta-feature can yield multiple measures for a given data set, depending

on whether it is computed on all data set features at once or a subset of them. For example, when using the correlation meta-feature, which calculates the correlation between the features and the target of a data set, the extraction must happen for each feature, thus yielding multiple values. Applying summarization techniques to reduce these measures to a single dimension is essential, making it a data set measure instead of a feature measure. Mean and standard deviation are two standard summarization functions commonly used in the literature.

The subset $em_r$ (Equation 4.3) contains metadata about the classifier ensemble $r$ from a repository entry $m_{t,r}$. The classifier ensemble metadata includes the ensemble configuration $ec_r$, with its algorithm and hyperparameters, as well as the performance $perf_r$ and diversity $div_r$ metrics from the ensemble $r$ on the task $t$. Multiple ensembles can be generated for a task $t$, each with its own repository entry. Ensuring the ensembles from a task $t$ have a fundamentally different structure is crucial for Meta-CLEO to succeed. This includes ensembles of different sizes, types of classifiers, and decision fusion methods. Nearly identical ensembles in the repository would add more processing but less value to the meta-learner result. The ensemble generation process for the tasks to form the repository can be made manually by an ML expert or automatically through AutoML tools. An example of a repository with three tasks and eight ensembles is available in Table 4.1. The repository entries for tasks 1 and 3 contain three ensemble solutions, while the entries for task 2 contain two. Ensuring that the task identifier is fixed for all repository entries containing ensemble solutions for the same task is key, as only ensemble solutions for the most similar task are evaluated in the online stage.

Table 4.1: Example of a repository in Meta-CLEO. The meta-features are the same for all classifier ensembles that solve the same task.

| $m_{t,r}$ | $tm_t$ | $em_r$ |
|---|---|---|
| $m_{1,1}$ | $\{id_1, mtf_1\}$ | $\{ec_1, perf_1, div_1\}$ |
| $m_{1,2}$ | $\{id_1, mtf_1\}$ | $\{ec_2, perf_2, div_2\}$ |
| $m_{1,3}$ | $\{id_1, mtf_1\}$ | $\{ec_3, perf_3, div_3\}$ |
| $m_{2,4}$ | $\{id_2, mtf_2\}$ | $\{ec_4, perf_4, div_4\}$ |
| $m_{2,5}$ | $\{id_2, mtf_2\}$ | $\{ec_5, perf_5, div_5\}$ |
| $m_{3,6}$ | $\{id_3, mtf_3\}$ | $\{ec_6, perf_6, div_6\}$ |
| $m_{3,7}$ | $\{id_3, mtf_3\}$ | $\{ec_7, perf_7, div_7\}$ |
| $m_{3,8}$ | $\{id_3, mtf_3\}$ | $\{ec_8, perf_8, div_8\}$ |

After the repository is built, normalization of the repository data is required as they contain values in different ranges. This is done to avoid having the meta-learner give more influence to a feature with a bigger range. So, the features passed to the meta-learner

are first re-scaled to the interval [0, 1] using the MinMax method described by Equation 4.4.

$$x_{\text{scaled}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \tag{4.4}$$

where:

$x_{\text{scaled}}$ is the re-scaled value of $x$

$x$ is the original value

$x_{\text{min}}$ is the minimum value for that feature in the data set

$x_{\text{max}}$ is the maximum value for that feature in the data set

The meta-learner training in Meta-CLEO involves training an algorithm to identify the repository task that resembles the most with a new task. This can be done through ML algorithms or association rules that take the meta-features from the new task and try to find the most similar instance available in the training data, i.e., the repository. In Meta-CLEO, the algorithm used within the meta-learner to predict the most similar task is the *k-Nearest-Neighbors* (kNN). kNN allows using simple and intuitive distance measures to calculate the similarity between instances. Leveraging kNN simplifies Meta-CLEO and makes it easier to explain the selection of the most similar task.

As the objective is to identify the single closest task from the new one, the kNN parameter is set to one. Later, in the online stage, one or more ensembles from the most similar task will be evaluated on the new task. The offline stage is completed when the repository is filled and the meta-learner is trained.

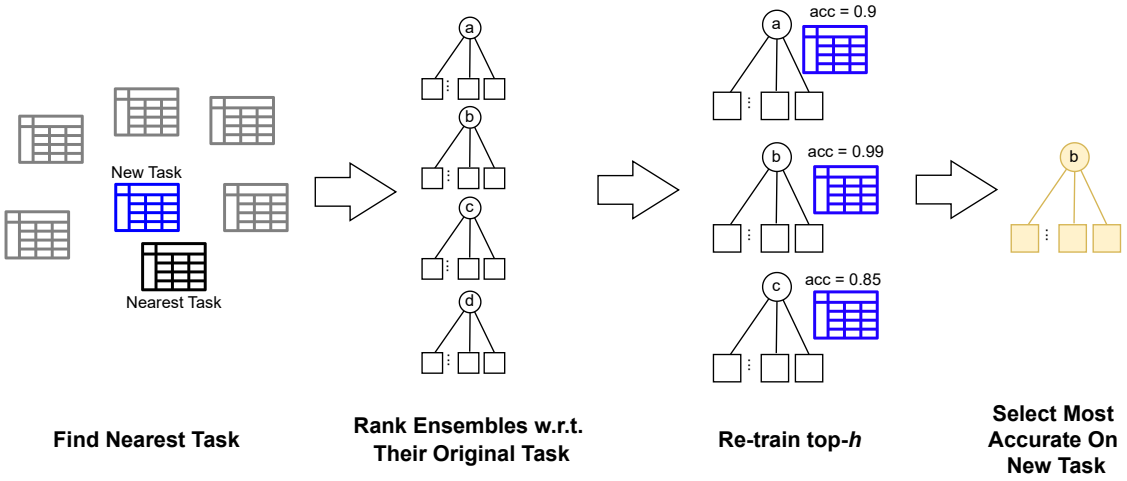
## 4.3 Online Stage

Once the user provides a new task, the online stage begins the ensemble creation process. As identified by the numbers in Figure 4.1, two major steps are required for the framework to return the new ensemble in the online stage: extracting meta-features from the new task in step 4 and the meta-learner execution in step 5.

Meta-feature extraction works the same way as in the offline stage, ensuring the same meta-features and summarization functions are used. The meta-features extracted from the train data set of the new task are then normalized before being prompted to

the meta-learner using the same normalizer from the meta-learner training process in the offline stage.

Figure 4.2: Meta-learner execution flow in Meta-CLEO. In the example, $h = 3$.




The normalized meta-features from the new task are the input to the meta-learner. An overview of the meta-learner execution can be seen in Figure 4.2. First, the meta-learner identifies the most similar task in the repository based on the meta-features of the new task. Then, it takes the ensembles of this task from the repository and ranks them based on their performance and diversity metrics on their original task. The balance between performance and diversity metrics used for ranking can vary, and experiments in Chapter 5 dive deeper into possible functions of these metrics and their impact on Meta-CLEO's final result. The top-*h* best-ranked ensembles are then re-trained on the new task.

During re-training, the ensembles are trained on the train data set of the new task. After re-training, the ensembles are evaluated on the test data set of the new task, and the most accurate ensemble is returned as the online stage final result.

One of the most distinguishing factors of Meta-CLEO is that it ranks the ensembles based on their predictive performance and diversity in original task and evaluates the top-$h$ ensembles, instead of only the one with the highest rank. By doing so, Meta-CLEO avoids limiting the final result to the single best ensemble from the original task, which may not be the top performer on the new task. Instead, the best ensemble for the new task may be the one with a level of diversity that leads to better generalization. The next Chapter details the impact of different ensemble ranking functions and $h$ values in Meta-CLEO.

# 5 EXPERIMENTAL EVALUATION

In this chapter, we evaluate the predictive performance of Meta-CLEO for new classification tasks based on the knowledge obtained from previously learned tasks. To perform this evaluation, tuning experiments were first carried out, and then the best Meta-CLEO configuration was compared with baselines. Meta-CLEO is a framework that uses meta-learning to create ensembles for new, unforeseen Machine Learning tasks while leveraging essential aspects of successful ensembles, such as diversity. The following sections describe the implementation developed in the scope of this thesis and three experiments built on top of it.

## 5.1 Implementation

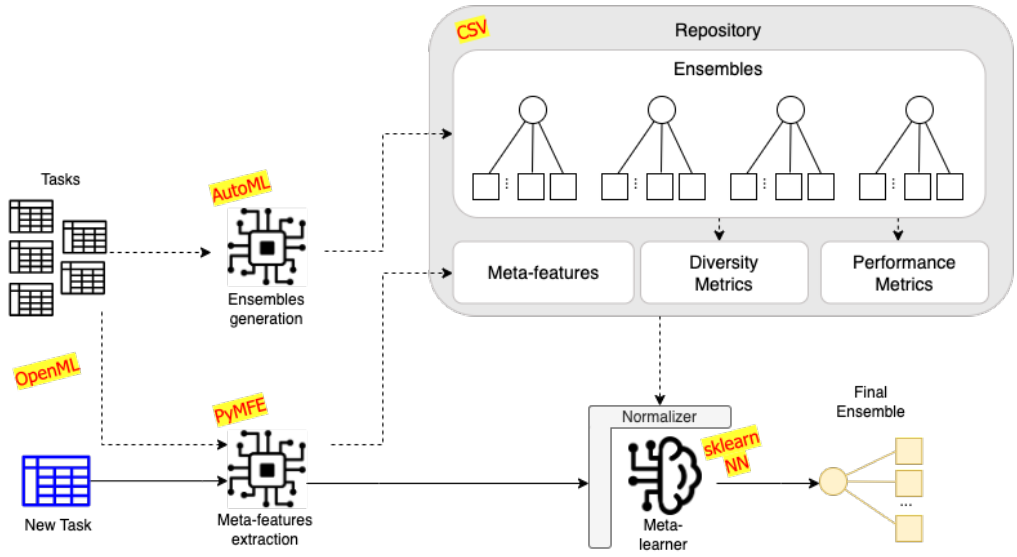
Implementing the Meta-CLEO framework was required to evaluate the framework and its effectiveness through experiments. Figure 5.1 illustrates the architecture, including technology-specific decisions taken for this implementation, which will be discussed throughout this section. The dotted lines indicate the offline stage, where the repository is built with metadata from various classification tasks and their ensemble solutions. The solid lines indicate the online stage, where a new task is received, and an ensemble solution is created by the meta-learner. In yellow, the technical decisions made during development are highlighted. The implementation was split into the two main components of meta-learning applications as listed in Section 2.3: the repository and the meta-learner. The repository creation includes the generation of the ensembles for the various tasks and their meta-feature extraction, while the meta-learner identifies the most similar task, ranks the ensembles, and re-trains them.

### 5.1.1 Setup

Meta-CLEO's implementation was built in an OpenStack m1.large virtual machine with four vCPUs and 8192 MB of RAM. The virtual machine was running Ubuntu Bionic 18.04 as the operating system and was not running anything besides the Meta-CLEO-related programs.

Python 3.8 was used for the implementation. The AutoML tool from Vogges-

Figure 5.1: Meta-CLEO high-level architecture including implementation decisions.




berger (2023) was used to generate the ensembles from the repository, leveraging heterogeneous base classifiers, various decision fusion mechanisms, and optimizing diversity. In addition, the following Python libraries were used: OpenML version 0.14.1 was used to download OpenML tasks and their datasets, PyMFE (ALCOBACA et al., 2020) version 0.4.2 was used to extract meta-features from the data sets, Pandas (MCKINNEY, 2010) version 1.4.0 was used to manipulate the data sets, and Scikit-learn (PEDREGOSA et al., 2011) version 1.0.2 was used as the backing ML library for the meta-learner. The exact same setup was used later during the experiments.

### 5.1.2 Repository

Building the metadata repository is one of the main challenges of creating a meta-learning framework. The quality of the data in the repository is essential as it directly impacts the efficacy of the meta-learner. This is known as the *garbage in, garbage out* concept in ML. Hence, it is relevant to build a metadata repository that contains many different tasks and ensemble solutions.

In the implementation of Meta-CLEO, 74 classification tasks were chosen from the OpenML[1] repository. These tasks include both real-world and synthetic data. The OpenML repository provides the data set as well as a pre-defined train and test split for each task. The same tasks have been used in similar literature, such as Auto-Sklearn. The selection of a large number of tasks aims to increase the likelihood of finding a

[1]Available at <https://openml.org/>. Last access on 31/05/2024.

Table 5.1: Algorithms supported by Voggesberger (2023).

| Usage | Algorithms |
|---|---|
| Base Classifiers | Adaboost, Bernoulli Naive Bayes, Decision Tree, Extremely Randomized Trees, Gaussian Naive Bayes, Gradient Boosting, KNN, Linear Discriminant Analysis, Linear Support Vector Machine, Multinomial Naive Bayes, Passive Aggressive, Quadratic Discriminant Analysis, Random Forest, Stochastic Gradient Descent |
| Decision Fusion | Simple Average, Behaviour Knowledge Space, Borda Count, Cosine Similarity, Decision Template, Dempster Shafer, KNN, Macro Majority Vote, Micro Majority Vote, Maximum Likelihood Estimator, Naive Bayes and Weighted Voting |

similar task to the one provided by the user, leading to more accurate results and a better experience for the framework users. The full list of selected tasks, as well as some of their characteristics, can be found in Appendix C.

With so many tasks and the desire to generate multiple ensemble solutions for each one of them, the automation of the ensemble creation process was required. To achieve this, an AutoML system was used, which reduced the overall time consumed in this step. The chosen tool for this was the one from Voggesberger (2023), as it empowers the generation of ensembles with high variability, including heterogeneous base classifiers and a myriad of decision fusion mechanisms. The full list of supported base classifiers and decision fusion mechanisms is available in Table 5.1. In addition to the data set, the AutoML system requires further input, such as time limits for the base classifiers and the decision fusion mechanism optimizations, and the desired ensemble size.

For Meta-CLEO's implementation, the optimization time was fixed to three hours for the base classifier plus three hours for the decision fusion optimization. These values were selected primarily due to the time constraints of a Master's Thesis. This means that, for each repository entry, 6 hours were dedicated to finding the corresponding ensemble solution. To allow for reproducibility and keep it feasible to execute within the scope of this thesis, four different ensemble sizes were used, those being 5, 10, 15, and 20. In addition, to provide variability to the results, each AutoML parameter combination of task and ensemble size was executed three times with randomly generated seeds. In the end, for each task from the repository, 12 executions of the AutoML system were run to generate the classifier ensembles. This means a total of $74 * 12 * 6 = 5.328$ hours dedicated to building the repository content.

Furthermore, the AutoML tool also calculates performance and diversity metrics. The performance metrics that compose Meta-CLEO's repository were accuracy and balanced accuracy. The diversity metrics used were disagreement and double fault. The experiments discuss the use of these metrics and their impact on the meta-learner.

Lastly, for the extraction of the meta-features (cf. Section 4.2), the library PyMFE was used. PyMFE offers more than 90 measures, including all the meta-feature groups discussed in Section 2.3. In Meta-CLEO, all but the ones that require model generation, such as landmarking or clustering, were used. This decision allowed the framework to fasten its online stage run time when meta-features must be extracted from the new, incoming task. Table 5.2 shows a fraction of the meta-features available in Meta-CLEO's implemented repository and their respective meta-feature groups. The exhaustive list of meta-features utilized in the implementation is available in Appendix B.

Table 5.2: Some meta-features used by Meta-CLEO. The complete table is available in Appendix B.

| Category | Meta-Feature |
|---|---|
| Simple | |
| | Number of classes (MICHIE et al., 1995) |
| | Frequency of each class (LINDNER; STUDER, 1999) |
| | Number of instances (MICHIE et al., 1995) |
| | Number of features (MICHIE et al., 1995) |
| | Ratio of number of instances to features (KUBA et al., 2002) |
| Statistical | |
| | Canonical correlation between the features and the target (KALOUSIS, 2002) |
| | Absolute correlation for each feature pair (CASTIELLO; CASTELLANO; FANELLI, 2005) |
| | Absolute covariance for each feature pair (CASTIELLO; CASTELLANO; FANELLI, 2005) |
| | Variance for each feature (CASTIELLO; CASTELLANO; FANELLI, 2005) |

| | |
|---|---|
| Info Theoretic | Noisiness of the features (MICHIE et al., 1995) |
| | Mutual information between feature and target (MICHIE et al., 1995) |
| | Joint entropy between feature and class (MICHIE et al., 1995) |
| | Equivalent attributes (MICHIE et al., 1995) |
| | Target's Shannon entropy (MICHIE et al., 1995) |
| | Features' Shannon entropy (MICHIE et al., 1995) |

Table 5.3 shows a fraction of the repository built for the experiments. The repository is a CSV file that contains, for each generated ensemble of each task, the metadata for the task and the ensemble solution. The first column is the task identifier, followed by the meta-features in columns 2-8. Column 9 contains the name of the ensemble configuration file, followed by the performance metrics of the ensemble on the respective task in columns 10-13 and the diversity metrics in columns 14-15. The ensemble configuration file contains what is required to re-create the ensemble, including the hyper-parameter values. As the meta-features characterize the data set of the task, and many ensembles are generated for each task, the rows with ensembles from the same task will have the same task ID and meta-feature values, as can be seen in rows 1-5, 6-14, 15-19. This repository is then used to train the meta-learner, detailed in the next subsection.

### 5.1.3 Meta-learner

The meta-learner is used to identify the most similar task available in the repository compared to the new one given by the user, perform ensemble ranking, re-train the top-$h$ ensembles, and return the most accurate ensemble configuration for a new task. The algorithm used for the meta-learner to implement Meta-CLEO is the *Nearest Neighbors*, from the Python library Scikit-learn. The parameter $k$, which represents the number of neighbors, is set to one as we want to find the most similar task available in the repository. Additionally, as the number of instances in the data set is rather small, brute force can be used to calculate the distance between all instances in the data set and the new one. This

Table 5.3: Excerpt from Meta-CLEO's repository. Some columns were omitted due to the overall table size.

| Task ID | Class Entr. | Cov. Mean | Kurtosis Mean | # Attr. | # Bin. | # Cat. | # Class | Ensemble Config. File | Acc. | Balanced Acc. | Prec. Micro | Prec. Macro | Disag. | Doublef. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 279 | 0.931 | 0.031 | -1.742 | 9 | 0 | 9 | 2 | 279_ens-size-15_seed-853 | 0.975 | 0.972 | 0.975 | 0.975 | 0.017 | 0.966 |
| 279 | 0.931 | 0.031 | -1.742 | 9 | 0 | 9 | 2 | 279_ens-size-5_seed-358 | 0.997 | 0.997 | 0.997 | 0.664 | 0.075 | 0.992 |
| 279 | 0.931 | 0.031 | -1.742 | 9 | 0 | 9 | 2 | 279_ens-size-15_seed-464 | 0.975 | 0.970 | 0.975 | 0.651 | 0.040 | 0.973 |
| 279 | 0.931 | 0.031 | -1.742 | 9 | 0 | 9 | 2 | 279_ens-size-10_seed-155 | 0.997 | 0.997 | 0.997 | 0.996 | 0.015 | 0.994 |
| 279 | 0.931 | 0.031 | -1.742 | 9 | 0 | 9 | 2 | 279_ens-size-20_seed-496 | 0.975 | 0.975 | 0.975 | 0.972 | 0.111 | 0.991 |
| 288 | 1.585 | 0.227 | -0.199 | 40 | 0 | 0 | 3 | 288_ens-size-5_seed-991 | 0.858 | 0.858 | 0.858 | 0.858 | 0.128 | 0.906 |
| 288 | 1.585 | 0.227 | -0.199 | 40 | 0 | 0 | 3 | 288_ens-size-10_seed-848 | 0.873 | 0.873 | 0.873 | 0.656 | 0.095 | 0.901 |
| 288 | 1.585 | 0.227 | -0.199 | 40 | 0 | 0 | 3 | 288_ens-size-15_seed-2 | 0.875 | 0.875 | 0.875 | 0.657 | 0.115 | 0.904 |
| 288 | 1.585 | 0.227 | -0.199 | 40 | 0 | 0 | 3 | 288_ens-size-5_seed-101 | 0.873 | 0.873 | 0.873 | 0.873 | 0.091 | 0.900 |
| 288 | 1.585 | 0.227 | -0.199 | 40 | 0 | 0 | 3 | 288_ens-size-20_seed-66 | 0.872 | 0.872 | 0.872 | 0.872 | 0.124 | 0.903 |
| 288 | 1.585 | 0.227 | -0.199 | 40 | 0 | 0 | 3 | 288_ens-size-5_seed-451 | 0.868 | 0.868 | 0.868 | 0.868 | 0.122 | 0.905 |
| 288 | 1.585 | 0.227 | -0.199 | 40 | 0 | 0 | 3 | 288_ens-size-15_seed-892 | 0.862 | 0.863 | 0.862 | 0.862 | 0.131 | 0.901 |
| 288 | 1.585 | 0.227 | -0.199 | 40 | 0 | 0 | 3 | 288_ens-size-10_seed-997 | 0.867 | 0.867 | 0.867 | 0.651 | 0.139 | 0.905 |
| 288 | 1.585 | 0.227 | -0.199 | 40 | 0 | 0 | 3 | 288_ens-size-15_seed-36 | 0.865 | 0.865 | 0.865 | 0.865 | 0.134 | 0.903 |
| 2122 | 3.504 | 0.004 | 2.916 | 6 | 0 | 6 | 18 | 2122_ens-size-15_seed-64 | 0.771 | 0.791 | 0.771 | 0.800 | 0.109 | 0.808 |
| 2122 | 3.504 | 0.004 | 2.916 | 6 | 0 | 6 | 18 | 2122_ens-size-20_seed-690 | 0.840 | 0.818 | 0.840 | 0.814 | 0.193 | 0.870 |
| 2122 | 3.504 | 0.004 | 2.916 | 6 | 0 | 6 | 18 | 2122_ens-size-5_seed-278 | 0.795 | 0.792 | 0.795 | 0.806 | 0.139 | 0.838 |
| 2122 | 3.504 | 0.004 | 2.916 | 6 | 0 | 6 | 18 | 2122_ens-size-10_seed-494 | 0.822 | 0.819 | 0.822 | 0.829 | 0.231 | 0.859 |
| 2122 | 3.504 | 0.004 | 2.916 | 6 | 0 | 6 | 18 | 2122_ens-size-5_seed-532 | 0.833 | 0.798 | 0.833 | 0.797 | 0.142 | 0.873 |

is exactly the behavior of the library when the *algorithm* parameter is set to *auto*, in which the library decides the most appropriate algorithm based on the data set values.

In the online stage, the trained meta-learner first identifies the most similar repository task to the current input and then retrieves the ensembles and their performance and diversity metrics available for the closest task. These metrics are then used to rank the ensembles. After ranking, the top-*h* best-ranked ensemble configurations are selected and then trained and evaluated on the train and test data sets of the input task, respectively. The most accurate one is selected and returned as the final result of the framework execution.

## 5.2 Experiments

The Meta-CLEO meta-learner allows tuning of key configurations, such as the ranking function and the number of ensembles to be re-trained, in order to achieve improved results. In this section, we discuss two experiments that evaluate the impact of altering the above-mentioned configurations and another one comparing Meta-CLEO to two baselines. The experiments are defined as follows:

- *Experiment 1* evaluates the impact of varying the *h*-value, which defines how many of the best ensembles from the most similar task will be considered by the meta-learner when creating the ensemble for a new task.

- *Experiment 2* analyzes the impact of different functions of performance and diversity that can be used for ranking the ensembles from the most similar task.

- *Experiment 3* compares the best Meta-CLEO configuration identified in the previous experiments with two widely used ensemble creation techniques, Random Forest and Adaboost.

In all experiments, all tasks from the repository were used for evaluation. So, for each repository task, the task was removed from the repository, the framework was executed, and the evaluation results were collected. By doing so, we avoid requiring new tasks only for evaluation purposes.

### 5.2.1 Experiment 1: Re-training Multiple Ensembles

In this experiment, we evaluate the trade-off between expanding the re-training from only the best ensemble to the top-$h$ best ensembles from the most similar task, and the runtime complexity of the framework. The more ensembles are re-trained, the more training sessions are required, thus increasing runtime complexity. Due to the ranking of the ensembles, we believe there is no need to re-train all the available ensembles from the most similar task, as that would be costly for the framework, but rather a smaller amount of them while still achieving the best results. By evaluating the balance between these two factors, i.e., the number of ensembles to be re-trained and the predictive performance, we expect to identify a reasonable number of ensembles to be re-trained that balances the predictive accuracy and runtime complexity of Meta-CLEO.

To evaluate and find the balance between the number of ensembles to be re-trained and the predictive performance achieved, we executed Meta-CLEO with five different configurations, each with an increasing amount of ensembles from the most similar task being re-trained. The values selected for the number of ensembles to be re-trained were:

- *One*, indicating the result if only the best ensemble, i.e., the top 1 ranked ensemble from the most similar task, was re-trained.
- *Three*, indicating the result if the top 3 higher ranked ensembles based on their predictive accuracy on their original task were re-trained.
- *Six*, indicating the result if the top 6 higher ranked ensembles based on their predictive accuracy on their original task were re-trained.
- *Nine*, indicating the result if the top 9 higher ranked ensembles based on their predictive accuracy on their original task were re-trained.
- *Twelve*, indicating the result if *all* ensembles generated for the most similar task were re-trained. This configuration will always lead to the *best possible result* Meta-CLEO can provide, as all ensembles from the most similar task are considered.

These values were selected as they provide a reasonable level of granularity while not exhausting all possible alternatives, i.e., all values from 1 to 12, which would take longer to execute and evaluate. The results from the execution of Meta-CLEO with the configuration where all ensembles from the most similar task are considered, i.e., with twelve ensembles being re-trained, will be the baseline for this experiment as they always lead to

the best results Meta-CLEO can provide. These results are referenced as the *best possible Meta-CLEO* results.

Figure 5.2: Percentage of tasks evaluated that achieve the best accuracy possible by re-training top 1, 3, 6, 9, and 12 most accurate ensembles from the most similar task.

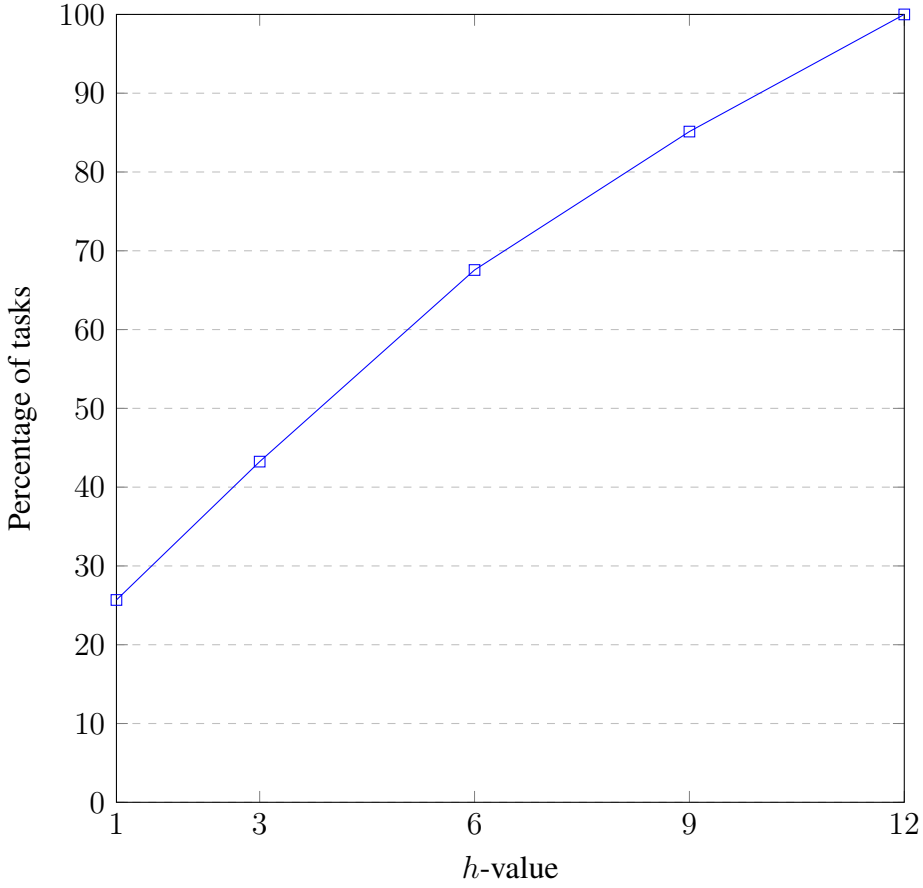



Figure 5.2 illustrates the percentage of tasks that achieve the best result Meta-CLEO can provide when re-training the top 1, 3, 6, 9, and 12 most accurate ensembles from the most similar task. The most accurate ensemble from the most similar task was the most accurate one for the new task 25.68% of the time. This means that, after ranking the ensembles from the most similar task based on predictive accuracy, re-training only the best one to create the ensemble for the new task has provided the best results Meta-CLEO could provide for 1 in every 4 tasks evaluated. From this information, we can infer that, for around 3 in every 4 tasks evaluated, the best possible result Meta-CLEO could provide was not the one with the highest predictive accuracy in the most similar task. So, to include the best possible ensemble that Meta-CLEO can provide in the results, the number of ensembles from the most similar task being re-trained was expanded from the single highest to the top-$h$ highest-ranked ensembles. In 43.24% of the cases, re-training the top 3 ensembles from the most similar task provided the best results. If expanded to the top 6 and 9, the percentage grows to 67.56% and 85.13%, respectively. In Figure 5.2,

Table 5.4: Average accuracy improvement gained by expanding the re-training to the top-$h$ highest-ranked ensembles, compared to the result obtained from re-training only the highest-ranked ensemble from the most similar task.

| $h = 3$ | $h = 6$ | $h = 9$ | $h = 12$ |
|---|---|---|---|
| +2.51pp | +3.31pp | +3.79pp | +4.39pp |

we can see an elbow that indicates where the increment of ensembles to be re-trained starts to have a smaller gain of accuracy. This happens after the top 6 mark and makes it visual that re-training the top 6 highest-ranked ensembles in the similar task is the configuration that provides the best trade-off between predictive accuracy and runtime complexity.

Table 5.4 shows the improved accuracy on average when the number of re-trained ensembles was increased. Expanding the re-training from the top 1 to the top 3 and 6 increased the accuracy by an average of 2.51 and 3.31 pp, as seen in the first and second columns, respectively. Expanding to re-train all ensembles, represented by the last column of the table, brings the best improvements. This is expected as it will always include the ensemble from the most similar task that provides the best results for the new task. So, taking into consideration that we desire to keep the number of training sessions as low as possible, leveraging the top 6 ensembles from the most similar task is prominent as the average accuracy improvement decreases for the top 9 and 12, i.e., if the average accuracy improvement grows 0.8 pp between the top 3 and 6, it stays lower than this value between top 6 and 9, and top 9 and 12. In addition, the T-test (ANDERSON; FINN, 2012) shows that the gain from expanding the re-training from the top 1 to the top 6 is statistically significant because the two-tailed p-value ($p = 0.008$) is lower than the significance coefficient adopted ($\alpha = 0.05$).

Table 5.5: Predictive accuracy of the Meta-CLEO results for some of the evaluated tasks in all configurations. The best result for each task is highlighted in bold.

| OpenML Task ID | $h = 1$ | $h = 3$ | $h = 6$ | $h = 9$ | $h = 12$ |
|---|---|---|---|---|---|
| 279 | 0.8354 | 0.8354 | 0.8671 | 0.9399 | **0.9462** |
| 288 | 0.8564 | 0.8618 | 0.8618 | **0.8630** | **0.8630** |
| 75131 | 0.9182 | **0.9303** | **0.9303** | **0.9303** | **0.9303** |
| 75180 | **0.9182** | **0.9182** | **0.9182** | **0.9182** | **0.9182** |
| 75187 | 0.9762 | 0.9771 | **0.9820** | **0.9820** | **0.9820** |
| 75235 | 0.9467 | 0.9467 | **0.9567** | **0.9567** | **0.9567** |
| 146596 | 0.9465 | 0.9465 | 0.9519 | 0.9519 | **0.9679** |
| 166950 | 0.8364 | **0.9152** | **0.9152** | **0.9152** | **0.9152** |
| 166958 | **0.9946** | **0.9946** | **0.9946** | **0.9946** | **0.9946** |
| 189863 | 0.8571 | 0.8571 | 0.8668 | 0.8668 | **0.8707** |
| 189881 | 0.9102 | 0.9102 | 0.9102 | **0.9122** | **0.9122** |
| 189902 | **0.9683** | **0.9689** | **0.9689** | **0.9689** | **0.9689** |

Table 5.5 displays the predictive accuracy from running Meta-CLEO with the 5 different configurations. For tasks 75180, 166958, and 189902, the best ensemble for the task is the best ensemble from the most similar task in the repository. This is because the distance to their nearest neighbor is small, i.e., lower than 0.15. As the most similar task and the new task are relatively alike, meta-learning prevails, and the best ensemble from the original task performs best in the new task. On the other hand, for tasks 279, 146596, and 189863, which required all ensembles from the most similar task to be re-trained in order to find the most accurate one, the distances to their nearest neighbor are above 0.35. Having a higher distance to the original task worsens the meta-learning process, as the ensemble was built originally for a task that is not so similar to the new one. Hence, ranking makes little difference, and the best ensemble for the new task could be ranked anywhere. This scenario would likely benefit more from an optimization step built on top of the resulting ensemble, fine-tuning the created ensemble to the new task.

### 5.2.2 Experiment 2: Leveraging Diversity Within the Meta-learner

Diversity has a key influence on building a successful ensemble. In this experiment, we explore the use of ensemble diversity metrics to rank classifier ensembles from the most similar task to accelerate finding the best result Meta-CLEO can provide for a task. As seen in Experiment 1, the best ensemble on the new task, i.e., the ensemble from the most similar task that generalized best on the new task, is not always the highest-ranked based on accuracy. Hence, we expect that, by including ensemble diversity metrics in the ranking function, Meta-CLEO will be able to provide the best result for new tasks with fewer ensembles being re-trained.

To assess this, we used the same configurations from Experiment 1, i.e., re-training the top 1, 3, 6, 9, and 12 highest-ranked ensembles from the most similar task, but now including an ensemble diversity metric in the ranking function. The diversity metric used to rank the classifier ensembles was *double fault*. Double fault was selected as it is simple and easy to understand and interpret. Additionally, the AutoML system of choice optimizes the ensembles with respect to double fault. Double fault is a pairwise ensemble diversity metric that measures the probability of the classifiers making wrong predictions for the same instance. Since double fault is a pairwise metric, to compute the double fault metric for the entire ensemble, the combination of all pairs of classifiers was generated, their double fault metric was calculated, and the average double fault from all combina-

tions was defined as the overall ensemble double fault metric. The higher the diversity, the lower the double fault measure. In this experiment, $1 - doublefault$ was used in all calculations to standardize it to interpret the higher, the better. The ranking functions utilized were: only the value of accuracy, only the value of double fault, and the average between accuracy and $1 - doublefault$, formulated in Equation 5.1.

$$\frac{accuracy + (1 - doublefault)}{2} \tag{5.1}$$

Figure 5.3: Percentage of tasks evaluated that achieve the best accuracy possible by re-training top 1, 3, 6, 9, and 12 highest ranked ensembles from the most similar task based on three different ranking functions.

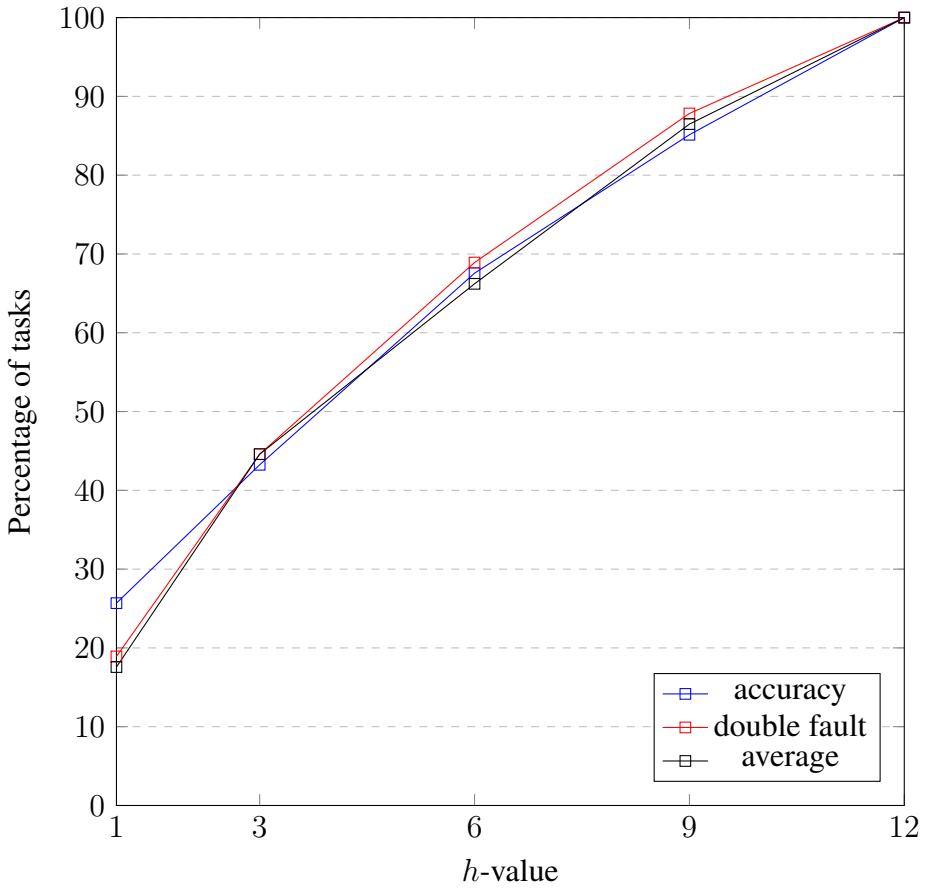


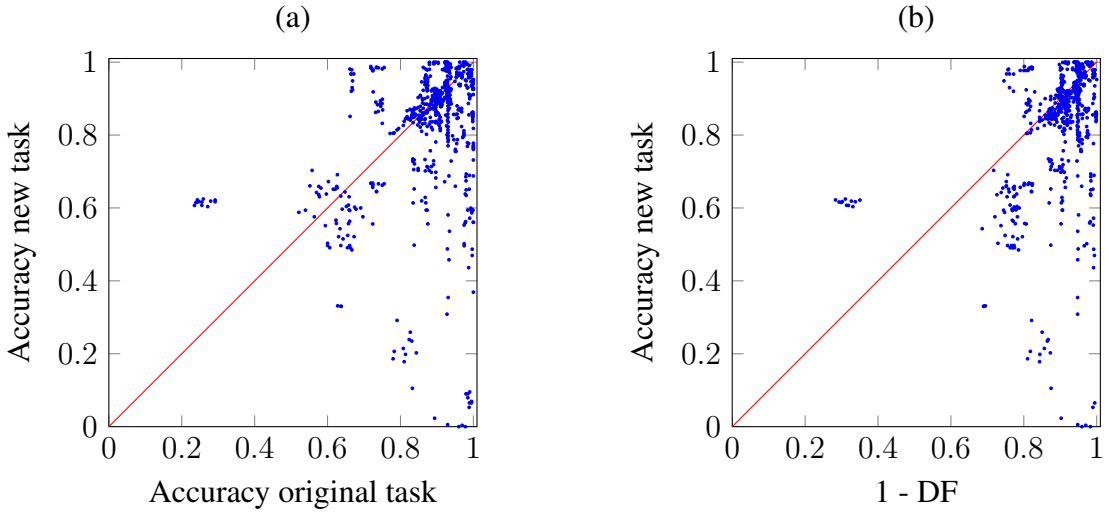

Figure 5.3 depicts how the ensemble that achieved the highest predictive performance for the new task was ranked in the original task based on the three different ranking functions. If only the top 1 highest ranked ensemble on the original task was re-trained, ranking on the accuracy value only provides the best possible ensemble Meta-CLEO can provide most of the time, if compared to ranking by double fault only and the average between accuracy and double fault. However, when re-training multiple ensembles, ranking by double fault yields better results than accuracy alone, and often better results than

Table 5.6: Distribution of the best ensemble from each task among the various rankings. The best result for each value of $h$ is highlighted in bold.

| Ranking Function | $h = 1$ | $h = 3$ | $h = 6$ | $h = 9$ | $h = 12$ |
|---|---|---|---|---|---|
| Accuracy | | | | | |
| Number of tasks | **19** | 32 | 50 | 63 | **74** |
| % | **25.68** | 43.24 | 67.56 | 85.13 | **100** |
| Doublefault | | | | | |
| Number of tasks | 14 | **33** | **51** | **65** | **74** |
| % | 18.92 | **44.59** | **68.91** | **87.83** | **100** |
| Average | | | | | |
| Number of tasks | 13 | **33** | 49 | 64 | **74** |
| % | 17.57 | **44.59** | 66.21 | 86.48 | **100** |

the average between accuracy and double fault. Table 5.6 lists the values from Figure 5.3, allowing for a more detailed analysis. When re-training the top 3 highest ranked ensembles, using only double fault or the average between accuracy and double fault as the ranking function provides the best result Meta-CLEO can provide 44.59% of the time, 1.35pp more than if only accuracy was used for ranking. Expanding to the top 6 and top 9, ranking by double fault only yields results better than ranking by accuracy only or the average between accuracy and double fault. Hence, when re-training multiple ensembles, using a diversity metric to rank the ensembles from the most similar task provides the best result more frequently than the other ranking functions that include the accuracy of the ensemble on the original task.

Figure 5.4: Correlation between accuracy and double fault from the ensemble in the original task and accuracy on the new task. The red line illustrates a perfect positive correlation (for reference).




This outcome aligns with the idea that diversity measures provide better indications with respect to the generalization capacity of the ensembles for a new task than the

predictive performance they had on the original task. Figures 5.4a and 5.4b display the correlation between the accuracy and the double fault metrics on the original task versus the accuracy of the re-trained ensemble on the new task, respectively. Accuracy in the original task and accuracy in the new task have a Pearson correlation coefficient of 0.29, which does not account for any correlation. On the other hand,$1 - double fault$ and accuracy on the new task have a Pearson correlation of 0.35, which is 0.06 higher and indicates a weak correlation between the metrics. By having a stronger correlation, the diversity metric demonstrates its superiority for ranking the ensembles before re-training.

### 5.2.3 Experiment 3: Comparison With Other Tools

In this experiment, we compare Meta-CLEO with two well-known ensemble algorithms that a novice user could use to solve a classification task. We expect Meta-CLEO to perform significantly best, as it creates ensembles for new tasks based on previous knowledge obtained in training similar tasks, i.e., by leveraging meta-learning.

The ensemble algorithms selected for comparison were Random Forest[2] and AdaBoost[3], as they are very common, well-established, and still widely used to solve ML tasks. All ensembles were trained using the default algorithm parameters provided by the library, as an inexperienced user would do. For Meta-CLEO, the configuration selected was $h = 6$ due to the best trade-off between runtime complexity and predictive accuracy, discussed in Experiment 1. The ranking function was set to the double fault metric, as it provided the best results in Experiment 2. In addition, re-training all the ensembles available from the most similar task was also evaluated. By doing so, we can explore Meta-CLEO in a scenario where the user is willing to spend more time on training to achieve better results.

Table 5.7 shows the percentage of tasks evaluated in which Meta-CLEO configurations performed better, worse, or equal to Random Forest and AdaBoost. When re-training only the top 6 highest-ranked ensembles from the most similar task, i.e., when $h = 6$, Meta-CLEO provides better results than Random Forest and AdaBoost for 70.27% and 94.59% of the tasks, respectively. If we expand to re-train all ensembles from the most similar task, i.e., when $h = 12$, Meta-CLEO performs better than Random For-

[2]<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier>

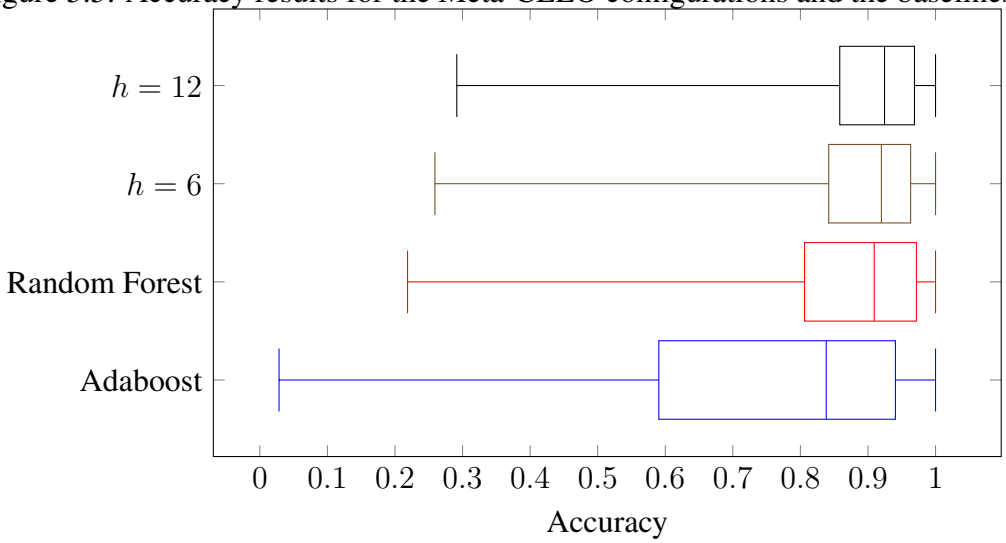
[3]<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#sklearn.ensemble.AdaBoostClassifier>

Table 5.7: Percentage of tasks in which Meta-CLEO configurations perform better, worse, or equal the baselines.

| Algorithm | $h = 6$ | $h = 12$ |
|---|---|---|
| Random Forest | | |
| Meta-CLEO performs best | 70.27% | 75.67% |
| Meta-CLEO performs worst | 24.32% | 18.91% |
| Meta-CLEO performs equal | 5.4% | 5.4% |
| AdaBoost | | |
| Meta-CLEO performs best | 94.59% | 95.94% |
| Meta-CLEO performs worst | 4.05% | 2.7% |
| Meta-CLEO performs equal | 1.35% | 1.35% |

est and Adaboost for 75.67% and 95.94% of the tasks, respectively. By re-training all ensembles from the most similar task, the resulting ensembles provide better results for tasks that were previously worse than the baselines. This is visible as the number of tasks in which Meta-CLEO performed worse than the baseline decreased from 24.32% to 18.91% for Random Forest and from 4.05% to 2.7% for AdaBoost. Considering that we are adding twice the runtime complexity to the execution of Meta-CLEO, having an improvement for only $75.67\% - 70.27\% = 5.4\%$ of the tasks for Random Forest and $95.94\% - 94.59\% = 1.35\%$ of the tasks for AdaBoost may be considered not worth it by the end user, thus making the Meta-CLEO $h = 6$ configuration the one to choose.

Figure 5.5: Accuracy results for the Meta-CLEO configurations and the baselines.




The boxplot from Figure 5.5 shows the improvements in accuracy when comparing the Meta-CLEO solutions to the baselines. AdaBoost is the worst performer overall, considering baselines and Meta-CLEO configurations. The interquartile range (IQR) for Random Forest was 0.165, while Meta-CLEO configurations $h = 6$ and $h = 12$ had an IQR of 0.122 and 0.11, respectively. The lower IQR in Meta-CLEO configurations

Table 5.8: Mean and median of the number of instances in tasks where Meta-CLEO and Random Forest performed best.

| Metric | Random Forest | Meta-CLEO |
|---|---|---|
| Mean | 19626 | 5258 |
| Median | 7708 | 1545 |

depicts the lower variability among their results.

On average, when the Meta-CLEO ensembles performed best than Random Forest, the improvements in accuracy provided by the Meta-CLEO implementations with $h = 6$ and $h = 12$ were of 3 pp and 3.3 pp. On the other hand, when Random Forests performed best, they were 6 pp and 4.1 pp more accurate than their Meta-CLEO counterparts with $h = 6$ and $h = 12$, respectively. This happens since Random Forests are known to work well under some circumstances, such as when the amount of data is bigger. This is the case for the tasks where Random Forest outperformed the best results Meta-CLEO could provide, i.e., when $h = 12$. Table 5.8 indicates this by showing that the mean number of instances from the tasks where Random Forest performed best is 19626, versus 5258 for tasks where Meta-CLEO performed best. The median values of 7708 and 1545 for Random Forest and Meta-CLEO, respectively, also reinforce this understanding.

The statistical test was made to check for statistical significance between the results obtained from the two Meta-CLEO configurations and the baselines. Statistical significance was identified between Meta-CLEO with $h = 6$ and $h = 12$ and AdaBoost. For Random Forest, however, statistical significance was only found when $h = 12$, where $p = 0.04$. Even though there is no statistical significance when comparing the results from Meta-CLEO with $h = 6$ and Random Forests, it is important to mention that the ensemble provided by Meta-CLEO can be later optimized. This can be done by, e.g., having Meta-CLEO as a first step in an AutoML tool that would further optimize the result provided by Meta-CLEO, thus yielding better results.

## 5.3 General Analysis of the Results

This chapter presented three experiments held on top of an implementation of Meta-CLEO. The first experiment evaluated the impact that re-training multiple ensembles has on finding the best result Meta-CLEO can provide for a task. From this experiment, we highlighted that Meta-CLEO configuration where $h = 6$ provided the best balance between the runtime complexity added by re-training more ensembles and the best

result Meta-CLEO can provide. In Experiment 2, we included diversity in the ranking function so that we can achieve the best result Meta-CLEO can provide in more scenarios than by ranking based on accuracy. The results from this second experiment confirmed the positive impact that diversity measures can have on indicating the generalization potential of an ensemble solution.

To wind up, we compared the most prominent Meta-CLEO configuration previously identified, i.e., the one with $h = 6$ and double fault as the ranking function, with two baselines. Furthermore, the baselines were compared to the best result Meta-CLEO could provide, i.e., when all ensembles from the most similar task were re-trained. Both Meta-CLEO configurations proved to be statistically significantly better than AdaBoost, one of the baselines. For the second baseline, Random Forest, Meta-CLEO was able to provide better results overall, providing improved results for more than 70% of the evaluated tasks. However, only the configuration where $h = 12$ was proved statistically significantly better than Random Forest. This is the case as Random Forest shines in scenarios where the data set is larger. Nevertheless, the configuration of Meta-CLEO where $h = 6$ can be further leveraged in future work as an alternative to warm-start AutoML, which would later optimize the ensemble created by Meta-CLEO, leading to better results.

# 6 CONCLUSION

In this Thesis, a new framework called Meta-CLEO was presented, which leverages knowledge from previously trained ensembles and their associated tasks to solve new classification tasks more accurately for non-technical ML users. Meta-CLEO combines meta-learning with heterogeneous ensembles to achieve its goal. It identifies the most similar task in its repository, ranks the associated ensembles, and re-trains one or more to find the one that fits best with the new task. Furthermore, Meta-CLEO leverages ensemble diversity explicitly to obtain better results while re-training fewer models.

Experiments conducted compared Meta-CLEO with two widely adopted ensemble techniques, namely Random Forest and AdaBoost, and concluded that Meta-CLEO provides better or equal results more than 75% of the time. The impact of including explicit diversity for ranking classifier ensembles and re-training multiple ensembles was also analyzed. Results indicate that using diversity metrics to rank the ensembles to be re-trained on the new task improves performance for some tasks when multiple ensembles are evaluated.

Much potential has been identified for the next steps of this work. First, meta-feature selection can take place to identify the meta-features that better suit the meta-learner. In the meta-learner, leveraging materialized data of the predictions from the base classifiers in the original task can allow variations of the proposed approaches when creating the ensembles for the new task during the online stage, especially considering ensemble diversity. Also, evaluating ensembles from more tasks, and not just the most similar one, may lead to improved results.

Regarding extensibility, Meta-CLEO can be integrated and used as a first step in an AutoML pipeline, warm-starting the AutoML process and thus giving more time for the optimizer to work on more promising results by reducing the search space. As the current implementation had its repository generated through an AutoML tool, it is reasonable to think that adding an optimization layer on top of the result provided by Meta-CLEO can also increase efficacy. If the Meta-CLEO framework is used in isolation, having the repository generated by a group of ML experts could also provide better results.

Last, regarding the experiments and the implementation, memory optimizations can be performed when de-serializing and loading the available models into memory, allowing for faster re-training and evaluation. More time can also be given as a budget to the AutoML tool used to populate the repository, leading to even better ensembles and

possibly better overall experiment results. Additionally, the related work can be used as baselines for comparison. If they include optimization, it could be turned off to evaluate solely the warm-starting phase, keeping the comparison fair. Including ensemble creation time as a dimension in the evaluation can also provide a better indication of the trade-off between time and accuracy when comparing optimized with non-optimized ensemble creation.

## REFERENCES

ALANNE, K.; SIERLA, S. An overview of machine learning applications for smart buildings. **Sustainable Cities and Society**, v. 76, p. 103445, 2022. ISSN 2210-6707. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S2210670721007186>.

ALCOBACA, E. et al. Mfe: Towards reproducible meta-feature extraction. **Journal of Machine Learning Research**, v. 21, n. 111, p. 1–5, 2020. Available from Internet: <http://jmlr.org/papers/v21/19-348.html>.

ALI, S.; SMITH, K. A. On learning algorithm selection for classification. **Applied Soft Computing**, Elsevier, v. 6, n. 2, p. 119–138, 2006.

ALI, S.; SMITH-MILES, K. A. A meta-learning approach to automatic kernel selection for support vector machines. **Neurocomputing**, v. 70, n. 1, p. 173–186, 2006. ISSN 0925-2312. Neural Networks. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0925231206001056>.

ANDERSON, T. W.; FINN, J. D. **The new statistical analysis of data**. [S.l.]: Springer Science & Business Media, 2012.

AVCI, O. et al. A review of vibration-based damage detection in civil structures: From traditional methods to machine learning and deep learning applications. **Mechanical Systems and Signal Processing**, v. 147, p. 107077, 2021. ISSN 0888-3270. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0888327020304635>.

Ayoub Shaikh, T.; RASOOL, T.; Rasheed Lone, F. Towards leveraging the role of machine learning and artificial intelligence in precision agriculture and smart farming. **Computers and Electronics in Agriculture**, v. 198, p. 107119, 2022. ISSN 0168-1699. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0168169922004367>.

BENSUSAN, H.; GIRAUD-CARRIER, C. Discovering task neighbourhoods through landmark learning performances. In: SPRINGER. **European Conference on Principles of Data Mining and Knowledge Discovery**. [S.l.], 2000. p. 325–330.

BREIMAN, L. Bagging predictors. **Machine learning**, Springer, v. 24, p. 123–140, 1996.

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, p. 5–32, 2001.

BROWN, G. et al. Diversity creation methods: a survey and categorisation. **Information fusion**, Elsevier, v. 6, n. 1, p. 5–20, 2005.

CARUANA, R. et al. Ensemble selection from libraries of models. In: **Proceedings of the twenty-first international conference on Machine learning**. [S.l.: s.n.], 2004. p. 18.

CASTIELLO, C.; CASTELLANO, G.; FANELLI, A. M. Meta-data: Characterization of input features for meta-learning. In: SPRINGER. **International Conference on Modeling Decisions for Artificial Intelligence**. [S.l.], 2005. p. 457–468.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 785–794. ISBN 9781450342322. Available from Internet: <https://doi.org/10.1145/2939672.2939785>.

COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE Transactions on Information Theory**, v. 13, n. 1, p. 21–27, 1967.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. On meta-learning for dynamic ensemble selection. In: IEEE. **2014 22nd International Conference on Pattern Recognition**. [S.l.], 2014. p. 1230–1235.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. Meta-des.oracle: Meta-learning and feature selection for dynamic ensemble selection. **Information Fusion**, v. 38, p. 84–103, 2017. ISSN 1566-2535. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S156625351730101X>.

CRUZ, R. M. et al. Meta-des: A dynamic ensemble selection framework using meta-learning. **Pattern Recognition**, v. 48, n. 5, p. 1925–1935, 2015. ISSN 0031-3203. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0031320314004919>.

CUNNINGHAM, P.; CARNEY, J. Diversity versus quality in classification ensembles based on feature selection. In: SPRINGER. **European Conference on Machine Learning**. [S.l.], 2000. p. 109–116.

DALENOGARE, L. S. et al. The expected contribution of industry 4.0 technologies for industrial performance. **International Journal of production economics**, Elsevier, v. 204, p. 383–394, 2018.

DIETTERICH, T. G. Ensemble methods in machine learning. In: SPRINGER. **International workshop on multiple classifier systems**. [S.l.], 2000. p. 1–15.

ENGELS, R.; THEUSINGER, C. Using a data metric for preprocessing advice for data mining applications. In: CITESEER. **ECAI**. [S.l.], 1998. v. 98, p. 23–28.

FEURER, M.; HUTTER, F. Hyperparameter optimization. In: ____. **Automated Machine Learning: Methods, Systems, Challenges**. Cham: Springer International Publishing, 2019. p. 3–33. ISBN 978-3-030-05318-5. Available from Internet: <https://doi.org/10.1007/978-3-030-05318-5_1>.

FEURER, M. et al. Efficient and robust automated machine learning. **Advances in neural information processing systems**, v. 28, 2015.

FEURER, M.; SPRINGENBERG, J. T.; HUTTER, F. Using meta-learning to initialize bayesian optimization of hyperparameters. In: **MetaSel@ ECAI**. [S.l.: s.n.], 2014. p. 3–10.

FREUND, Y.; SCHAPIRE, R.; ABE, N. A short introduction to boosting. **Journal-Japanese Society For Artificial Intelligence**, JAPANESE SOC ARTIFICIAL INTELL, v. 14, n. 771-780, p. 1612, 1999.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. **Journal of Computer and System Sciences**, v. 55, n. 1, p. 119–139, 1997. ISSN 0022-0000. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S002200009791504X>.

FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. **The Annals of Statistics**, Institute of Mathematical Statistics, v. 29, n. 5, p. 1189 – 1232, 2001. Available from Internet: <https://doi.org/10.1214/aos/1013203451>.

GALAR, M. et al. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, IEEE, v. 42, n. 4, p. 463–484, 2011.

HANSEN, L. K.; SALAMON, P. Neural network ensembles. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 12, n. 10, p. 993–1001, 1990.

HE, H.; GARCIA, E. A. Learning from imbalanced data. **IEEE Transactions on Knowledge and Data Engineering**, v. 21, n. 9, p. 1263–1284, 2009.

HIRSCH, V.; REIMANN, P.; MITSCHANG, B. Data-driven fault diagnosis in end-of-line testing of complex products. In: IEEE. **2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)**. [S.l.], 2019. p. 492–503.

HO, T. K. The random subspace method for constructing decision forests. **IEEE transactions on pattern analysis and machine intelligence**, Ieee, v. 20, n. 8, p. 832–844, 1998.

HO, T. K.; HULL, J. J.; SRIHARI, S. N. Decision combination in multiple classifier systems. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 16, n. 1, p. 66–75, 1994.

HOTELLING, H. A generalized t test and measure of multivariate dispersion. In: UNIVERSITY OF CALIFORNIA PRESS. **Proceedings of the second Berkeley symposium on mathematical statistics and probability**. [S.l.], 1951. v. 2, p. 23–42.

KALOUSIS, A. **Algorithm Selection via Meta-Learning**. Thesis (PhD thesis) — University of Geneva, 2002. Available at <https://cui.unige.ch/~kalousis/papers/metalearning/PhdThesisKalousis.pdf>.

KALOUSIS, A.; HILARIO, M. Model selection via meta-learning: a comparative study. In: **Proceedings 12th IEEE Internationals Conference on Tools with Artificial Intelligence. ICTAI 2000**. [S.l.: s.n.], 2000. p. 406–413.

KALOUSIS, A.; THEOHARIS, T. Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. **Intelligent Data Analysis**, v. 3, n. 5, p. 319–337, 1999. ISSN 1088-467X. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S1088467X99000268>.

KHAMBRA, G.; SHUKLA, P. Novel machine learning applications on fly ash based concrete: An overview. **Materials Today: Proceedings**, v. 80, p. 3411–3417, 2023. ISSN 2214-7853. SI:5 NANO 2021. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S221478532105121X>.

KHAN, I. et al. A literature survey and empirical study of meta-learning for classifier selection. **IEEE Access**, v. 8, p. 10262–10281, 2020.

KO, A. H.; SABOURIN, R.; JR, A. S. B. From dynamic classifier selection to dynamic ensemble selection. **Pattern recognition**, Elsevier, v. 41, n. 5, p. 1718–1731, 2008.

KOHAVI, R.; WOLPERT, D. Bias plus variance decomposition for zero-one loss functions. In: **Proceedings of the Thirteenth International Conference on International Conference on Machine Learning**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996. (ICML'96), p. 275–283. ISBN 1558604197.

KÖPF, C.; IGLEZAKIS, I. Combination of task description strategies and case base properties for meta-learning. In: **Proceedings of the 2nd international workshop on integration and collaboration aspects of data mining, decision support and meta-learning**. [S.l.: s.n.], 2002. p. 65–76.

KÖPF, C.; TAYLOR, C.; KELLER, J. Meta-analysis: From data characterisation for meta-learning to meta-regression. In: **Proceedings of the PKDD-00 workshop on data mining, decision support, meta-learning and ILP**. [S.l.: s.n.], 2000.

KORDÍK, P.; ČERNỲ, J.; FRỲDA, T. Discovering predictive ensembles for transfer learning and meta-learning. **Machine learning**, Springer, v. 107, n. 1, p. 177–207, 2018.

KUBA, P. et al. Exploiting sampling and meta-learning for parameter setting for support vector machines. University of Sevilla, 2002.

KUNCHEVA, L. I. Diversity in classifier ensembles. In: ____. **Combining Pattern Classifiers**. John Wiley  Sons, Ltd, 2004. chp. 10, p. 295–327. ISBN 9780471660262. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1002/0471660264.ch10>.

LASI, H. et al. Industry 4.0. **Business & information systems engineering**, Springer, v. 6, p. 239–242, 2014.

LINDNER, G.; STUDER, R. Ast: Support for algorithm selection with a cbr approach. In: SPRINGER. **European Conference on Principles of Data Mining and Knowledge Discovery**. [S.l.], 1999. p. 418–423.

MA, Z. et al. Meta learning-based hybrid ensemble approach for short-term wind speed forecasting. **IEEE Access**, IEEE, v. 8, p. 172859–172868, 2020.

MAHESH, B. Machine learning algorithms-a review. **International Journal of Science and Research (IJSR).[Internet]**, v. 9, n. 1, p. 381–386, 2020.

MCKINNEY Wes. Data Structures for Statistical Computing in Python. In: WALT Stéfan van der; MILLMAN Jarrod (Ed.). **Proceedings of the 9th Python in Science Conference**. [S.l.: s.n.], 2010. p. 56 – 61.

MICHIE, D. et al. (Ed.). **Machine learning, neural and statistical classification**. USA: Ellis Horwood, 1995. ISBN 013106360X.

MONTEIRO, J. P. et al. Meta-learning and the new challenges of machine learning. **International Journal of Intelligent Systems**, v. 36, n. 11, p. 6240–6272, 2021. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1002/int.22549>.

PAVEL, Y.; SOARES, B. C. Decision tree-based data characterization for meta-learning. **IDDM-2002**, v. 111, 2002.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PILLAI, K. S. Some new test criteria in multivariate analysis. **The Annals of Mathematical Statistics**, JSTOR, p. 117–121, 1955.

POLIKAR, R. Ensemble based systems in decision making. **IEEE Circuits and Systems Magazine**, v. 6, n. 3, p. 21–45, 2006.

PROKHORENKOVA, L. et al. Catboost: unbiased boosting with categorical features. **Advances in neural information processing systems**, v. 31, 2018.

QUINLAN, J. R. Induction of decision trees. **Machine learning**, Springer, v. 1, p. 81–106, 1986.

REIF, M. et al. Automatic classifier selection for non-experts. **Pattern Analysis and Applications**, Springer, v. 17, p. 83–96, 2014.

RIJN, J. N. van et al. Having a blast: Meta-learning and heterogeneous ensembles for data streams. In: IEEE. **2015 ieee international conference on data mining**. [S.l.], 2015. p. 1003–1008.

RIVOLLI, A. et al. **Characterizing classification datasets: a study of meta-features for meta-learning**. 2019.

ROKACH, L. Ensemble-based classifiers. **Artificial intelligence review**, Springer, v. 33, p. 1–39, 2010.

ROY, S. N. On a heuristic method of test construction and its use in multivariate analysis. **The Annals of Mathematical Statistics**, Institute of Mathematical Statistics, v. 24, n. 2, p. 220–238, 1953.

SAGI, O.; ROKACH, L. Ensemble learning: A survey. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, Wiley Online Library, v. 8, n. 4, p. e1249, 2018.

SALAMA, M. A.; HASSANIEN, A. E.; REVETT, K. Employment of neural network and rough set in meta-learning. **Memetic Computing**, Springer, v. 5, p. 165–177, 2013.

SHARMA, R. et al. A systematic literature review on machine learning applications for sustainable agriculture supply chain performance. **Computers Operations Research**, v. 119, p. 104926, 2020. ISSN 0305-0548. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0305054820300435>.

SHIFAZ, A. et al. Ts-chief: A scalable and accurate forest algorithm for time series classification. **Data Min. Knowl. Discov.**, Kluwer Academic Publishers, USA, v. 34, n. 3, p. 742–775, may 2020. ISSN 1384-5810. Available from Internet: <https://doi.org/10.1007/s10618-020-00679-8>.

SHWARTZ-ZIV, R.; ARMON, A. Tabular data: Deep learning is not all you need. **Information Fusion**, Elsevier, v. 81, p. 84–90, 2022.

SILVA, R. A. D. et al. Automatic recommendation method for classifier ensemble structure using meta-learning. **IEEE Access**, v. 9, p. 106254–106268, 2021.

SMITH-MILES, K. A. Cross-disciplinary perspectives on meta-learning for algorithm selection. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 41, n. 1, p. 1–25, 2009.

THORNTON, C. 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**. [S.l.: s.n.], 2013. p. 847–855.

THRUN, S.; PRATT, L. Learning to learn: Introduction and overview. In: ____. **Learning to Learn**. Boston, MA: Springer US, 1998. p. 3–17. ISBN 978-1-4615-5529-2. Available from Internet: <https://doi.org/10.1007/978-1-4615-5529-2_1>.

TUMER, K.; GHOSH, J. Error correlation and error reduction in ensemble classifiers. **Connection science**, Taylor & Francis, v. 8, n. 3-4, p. 385–404, 1996.

VANSCHOREN, J. Meta-learning. In: ____. **Automated Machine Learning: Methods, Systems, Challenges**. Cham: Springer International Publishing, 2019. p. 35–61. ISBN 978-3-030-05318-5. Available from Internet: <https://doi.org/10.1007/978-3-030-05318-5_2>.

VOGGESBERGER, J. Optimierung von klassifikator-ensembles mit automl. In: **Workshop für Grundlagen von Datenbanken**. [S.l.: s.n.], 2023. p. 9.

VOGGESBERGER, J.; REIMANN, P.; MITSCHANG, B. Towards the automatic creation of optimized classifier ensembles. In: **ICEIS (1)**. [S.l.: s.n.], 2023. p. 615–622.

WIDROW, B.; LEHR, M. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. **Proceedings of the IEEE**, v. 78, n. 9, p. 1415–1442, 1990.

WILHELM, Y. et al. Pusion - a generic and automated framework for decision fusion. In: **2023 IEEE 39th International Conference on Data Engineering (ICDE)**. [S.l.: s.n.], 2023. p. 3282–3295.

WOLPERT, D. H. Stacked generalization. **Neural Networks**, v. 5, n. 2, p. 241–259, 1992. ISSN 0893-6080. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0893608005800231>.

# APPENDIX A — RESUMO EXPANDIDO

O aprendizado de máquina (*Machine Learning* - ML) nunca foi tão popular. Com sua crescente adoção e a evolução dos recursos de hardware e da computação em nuvem, cada vez mais áreas de pesquisa e indústrias estão expandindo seus investimentos em tecnologia para resolver seus desafios de domínio com ML. Porém, o desenvolvimento de um modelo com alto desempenho preditivo para resolver uma determinada tarefa é um processo complexo. Um algoritmo deve ser escolhido e seus hiperparâmetros devem ser ajustados. Para *ensembles*, o desafio é ainda maior, pois vários algoritmos devem ser escolhidos, cada um com seus próprios hiperparâmetros, e, adicionalmente, um mecanismo de fusão deve ser escolhido, que também pode precisar de ajuste de hiperparâmetros. Especialistas em aprendizado de máquina contam com seu conhecimento e experiência anterior na resoloução de várias tarefas para enfrentar esse desafio e desenvolver uma solução para uma nova tarefa. No entanto, para analistas novatos que ainda não têm experiência em ML, criar uma solução que resolva a tarefa com alta precisão pode se tornar um processo desafiador.

Uma abordagem que tem sido usada com sucesso para tornar ML mais acessível aos analistas iniciantes é o meta-aprendizado. O meta-aprendizado, também conhecido como aprender a aprender, é um subcampo de ML que aproveita os metadados coletados de tarefas resolvidas anteriormente e suas soluções de ML para aprender novas tarefas com mais eficiência e precisão (VANSCHOREN, 2019). Com o meta-aprendizado, é possível determinar quais algoritmos tiveram um bom desempenho em tarefas semelhantes e usar essas informações para produzir uma solução para uma nova tarefa que antes era desconhecida. Embora o meta-aprendizado tenha sido usado no contexto de tarefas de classificação e *ensembles* no passado (KHAN et al., 2020; SILVA et al., 2021), não foi encontrada nenhuma literatura que explorasse como *ensembles* de classificadores heterogêneos se comportam em um conjunto de tarefas para que um *ensemble* de classificadores com bom desempenho possa ser criado para uma nova tarefa.

Esta dissertação apresenta uma nova estrutura que combina o conhecimento passado do meta-aprendizado com a eficácia comprovada dos *ensembles* de classificadores, denominada Meta-CLEO (***Meta**-Learning for **Cl**assifier **E**nsemble **O**ptimization*). Ao mesclar *ensembles* com o meta-aprendizado, é possível aproveitar os aspectos que tornam um *ensemble* eficaz, como a diversidade, bem como os componentes do *ensemble*, como o tamanho, os classificadores base e o método de fusão, como metadados exclusivos

para o meta-aprendizado. A contribuição deste trabalho inclui:

- Uma nova maneira de combinar meta-aprendizado e *ensembles* heterogêneos para auxiliar usuários não técnicos de ML a encontrar uma solução de *ensembles* com bom desempenho para suas tarefas;
- A proposta e a avaliação de um sistema de classificação baseado no desempenho do *ensemble* e em métricas de diversidade que compõem *ensembles* de bom desempenho e reduzem a complexidade do tempo de execução; e
- A avaliação do Meta-CLEO em relação a duas técnicas de *ensembles* amplamente conhecidas e comumente usadas em ML: Random Forest e AdaBoost;

A arquitetura típica de uma aplicação de meta-aprendizado é composta, principalmente, por um repositório e um meta-aprendiz. O repositório armazena os metadados das tarefas aprendidas anteriormente, entituladas *meta-features*, os modelos de ML usados e seus resultados de avaliação. Como a criação do repositório pode ser demorada, ela ocorre *offline*. O repositório é então usado para treinar o meta-aprendiz, o algoritmo responsável por aprender a relação entre as *meta-features* e os resultados da avaliação do modelo para uma tarefa. O meta-aprendiz pode ser um algoritmo de ML, como o kNN, ou um sistema baseado em regras que, com base nas *meta-features* de uma nova tarefa, retorna um modelo com base nos dados do repositório. A predição do meta-aprendiz ocorre *online*, ou seja, durante o tempo de execução.

No Meta-CLEO, o repositório é criado treinando e armazenando diferentes *ensembles* aprendidos para cada tarefa do repositório na etapa *offline*. Os *ensembles* são compostos por classificadores de base heterogêneos e um método de fusão. Quando uma nova tarefa é recebida, o meta-aprendiz identifica a tarefa mais similar e classifica seus *ensembles* com base nas medidas de desempenho **e** diversidade. Em seguida, os *ensembles* melhor ranqueados na tarefa mais similar são treinados no conjunto de dados de treinamento da nova tarefa e avaliados no respectivo conjunto de dados de teste. Por fim, o *ensemble* com melhor desempenho na nova tarefa é selecionado como a saída do meta-prendiz.

O meta-aprendiz do Meta-CLEO permite o ajuste das principais configurações, como a função de ranqueamento e o número de *ensembles* a serem re-treinados, o valor-$h$, a fim de obter melhores resultados. Dois experimentos foram realizados para avaliar o impacto da variação dessas configurações e outro para comparar o Meta-CLEO a dois

*baselines*. A partir dos experimentos, destacamos que a configuração do Meta-CLEO em que $h = 6$ forneceu o melhor equilíbrio entre a complexidade da execução adicionada pelo re-treino de múltiplos *ensembles* e o melhor resultado que o Meta-CLEO pode fornecer. No Experimento 2, incluímos a diversidade na função de ranqueamento para que possamos obter o melhor resultado que o Meta-CLEO pode oferecer em mais cenários do que com o ranqueamento baseado somente em precisão. No último experimento, comparamos a configuração mais proeminente do Meta-CLEO identificada anteriormente, ou seja, aquela com $h = 6$ e diversidade considerada na função de ranqueamento, com dois *baselines*. Além disso, os *baselines* foram comparados com o melhor resultado que o Meta-CLEO poderia fornecer, ou seja, quando todos os *ensembles* da tarefa mais similar foram re-treinados. Ambas as configurações do Meta-CLEO demonstraram ser estatisticamente melhores do que o *AdaBoost*, um dos *baselines*. Para o segundo *baseline*, o *Random Forest*, o Meta-CLEO foi capaz de fornecer melhores resultados em geral, proporcionando melhores resultados para mais de 70% das tarefas avaliadas. No entanto, somente a configuração em que $h = 12$ se mostrou estatisticamente melhor do que o *Random Forest*. Isso acontece pois o *Random Forest* se destaca em cenários em que o conjunto de dados é maior.

Por fim, foi identificado um grande potencial para trabalhos futuros a partir deste trabalho. Primeiramente, pode ser concedido mais tempo como orçamento para a ferramenta AutoML utilizada para preencher o repositório criado para os experimentos, o que pode resultar em *ensembles* ainda melhores e, possivelmente, em uma melhor avaliação nos experimentos. Além disso, a seleção de *meta-features* pode ser feita para identificar as que melhor se adequam ao meta-aprendiz. Em segundo lugar, com relação à extensibilidade, o Meta-CLEO pode ser integrado e usado como primeira etapa em um pipeline de AutoML, inicializando o processo de AutoML e, assim, dando mais tempo para que o otimizador trabalhe em resultados mais promissores, reduzindo o espaço de pesquisa.

# APPENDIX B — COMPLETE TABLE OF META-FEATURES USED IN THE IMPLEMENTATION OF META-CLEO

Table B.1: Meta-features used by Meta-CLEO.

| Category | Meta-Feature |
|---|---|
| Simple | |
| | Number of classes (MICHIE et al., 1995) |
| | Frequency of each class (LINDNER; STUDER, 1999) |
| | Number of instances (MICHIE et al., 1995) |
| | Number of features (MICHIE et al., 1995) |
| | Ratio of number of instances to features (KUBA et al., 2002) |
| | Ratio of number of features to instances (KALOUSIS; THEOHARIS, 1999) |
| | Number of binary features (MICHIE et al., 1995) |
| | Number of categorical features (ENGELS; THEUSINGER, 1998) |
| | Number of numerical features (ENGELS; THEUSINGER, 1998) |
| | Ratio of numerical to categorical features (FEURER; SPRINGENBERG; HUTTER, 2014) |
| | Ratio of categorical to numerical features (FEURER; SPRINGENBERG; HUTTER, 2014) |
| Statistical | |
| | Canonical correlation between the features and the target (KALOUSIS, 2002) |
| | Absolute correlation for each feature pair (CASTIELLO; CASTELLANO; FANELLI, 2005) |
| | Absolute covariance for each feature pair (CASTIELLO; CASTELLANO; FANELLI, 2005) |
| | Variance for each feature (CASTIELLO; CASTELLANO; FANELLI, 2005) |
| | Eigenvalues of the covariance matrix (ALI; SMITH, 2006) |
| | Data set target gravity (ALI; SMITH, 2006) |

| | |
|---|---|
| | Median Absolute Deviation (ALI; SMITH, 2006) |
| | Geometric mean for each feature (ALI; SMITH-MILES, 2006) |
| | Harmonic mean for each feature (ALI; SMITH-MILES, 2006) |
| | Range of each feature (ALI; SMITH-MILES, 2006) |
| | Interquartile range for each feature (ALI; SMITH-MILES, 2006) |
| | Kurtosis for each feature (MICHIE et al., 1995) |
| | Homogeneity of covariances statistical test (MICHIE et al., 1995) |
| | Skewness of each feature (MICHIE et al., 1995) |
| | Lawley-Hotelling trace (HOTELLING, 1951) |
| | Maximum / Minimum value for each feature (ENGELS; THEUSINGER, 1998) |
| | Mean value for each feature (ENGELS; THEUSINGER, 1998) |
| | Median value for each feature (ENGELS; THEUSINGER, 1998) |
| | Standard deviation for each feature (ENGELS; THEUSINGER, 1998) |
| | Trimmed mean for each feature (ENGELS; THEUSINGER, 1998) |
| | Number of highly correlated feature pairs (SALAMA; HASSANIEN; REVETT, 2013) |
| | Sparsity measure for each feature (SALAMA; HASSANIEN; REVETT, 2013) |
| | Number of canonical correlations between each feature and target (LINDNER; STUDER, 1999) |
| | Wilks' lambda value (LINDNER; STUDER, 1999) |
| | Number of normally distributed features (KÖPF; TAYLOR; KELLER, 2000) |
| | Number of features with at least one outlier (KÖPF; IGLEZAKIS, 2002) |
| | Pillai's trace (PILLAI, 1955) |
| | Roy's largest root (ROY, 1953) |
| Info Theoretic | |
| | Noisiness of the features (MICHIE et al., 1995) |

| | |
|---|---|
| | Mutual information between feature and target (MICHIE et al., 1995) |
| | Joint entropy between feature and class (MICHIE et al., 1995) |
| | Equivalent attributes (MICHIE et al., 1995) |
| | Target's Shannon entropy (MICHIE et al., 1995) |
| | Features' Shannon entropy (MICHIE et al., 1995) |
| | Concentration coefficient between each feature and class (KALOUSIS; HILARIO, 2000) |
| | Concentration coefficient for each feature pair (KALOUSIS; HILARIO, 2000) |

# APPENDIX C — TASKS USED IN THE IMPLEMENTATION AND EVALUATION OF META-CLEO

Table C.1: All tasks used in Meta-CLEO and their key aspects.

| OpenML Task ID | Number of Features | Number of Instances | Number of Classes |
|---|---|---|---|
| 279 | 9 | 958 | 2 |
| 288 | 40 | 5000 | 3 |
| 2122 | 6 | 28056 | 18 |
| 3048 | 6 | 11183 | 2 |
| 3049 | 49 | 937 | 2 |
| 3053 | 6 | 556 | 2 |
| 75089 | 20 | 1000 | 2 |
| 75108 | 167 | 6598 | 2 |
| 75112 | 10 | 19020 | 2 |
| 75115 | 10935 | 1545 | 2 |
| 75121 | 10935 | 1545 | 2 |
| 75125 | 10935 | 1545 | 2 |
| 75126 | 10935 | 1545 | 2 |
| 75131 | 5 | 1000 | 2 |
| 75134 | 7 | 164860 | 11 |
| 75139 | 48 | 15000 | 2 |
| 75141 | 8 | 8192 | 2 |
| 75142 | 10 | 40768 | 2 |
| 75146 | 40 | 13750 | 2 |
| 75147 | 12 | 8192 | 2 |
| 75148 | 6 | 3107 | 2 |
| 75149 | 10 | 1000 | 2 |
| 75154 | 64 | 1600 | 100 |
| 75156 | 1776 | 3751 | 2 |
| 75171 | 8 | 8192 | 2 |

| | | | |
|---|---|---|---|
| 75176 | 8 | 20640 | 2 |
| 75180 | 50 | 1000 | 2 |
| 75185 | 14 | 6574 | 2 |
| 75187 | 20 | 7400 | 2 |
| 75199 | 25 | 1000 | 2 |
| 75210 | 4 | 8641 | 2 |
| 75219 | 14 | 14980 | 2 |
| 75223 | 6 | 28056 | 18 |
| 75233 | 21 | 8192 | 2 |
| 75235 | 24 | 5456 | 4 |
| 75236 | 256 | 1593 | 10 |
| 126021 | 14 | 9961 | 9 |
| 126024 | 5 | 15545 | 2 |
| 126030 | 561 | 10299 | 6 |
| 126031 | 40 | 5500 | 11 |
| 146574 | 60 | 600 | 6 |
| 146576 | 70 | 841 | 4 |
| 146586 | 4 | 1372 | 2 |
| 146592 | 100 | 1212 | 2 |
| 146593 | 10 | 583 | 2 |
| 146594 | 500 | 2600 | 2 |
| 146596 | 30 | 569 | 2 |
| 146597 | 1300 | 571 | 20 |
| 166866 | 100 | 500 | 2 |
| 166872 | 7 | 500 | 2 |
| 166905 | 20 | 506 | 2 |
| 166915 | 9 | 950 | 2 |
| 166931 | 5 | 500 | 2 |
| 166932 | 7 | 500 | 2 |
| 166950 | 10 | 500 | 2 |
| 166951 | 50 | 500 | 2 |
| 166956 | 4 | 559 | 2 |

| | | | |
|---|---|---|---|
| 166958 | 4 | 559 | 2 |
| 167085 | 1000 | 1600 | 2 |
| 167086 | 20 | 1600 | 2 |
| 167089 | 20 | 1600 | 2 |
| 189859 | 10304 | 575 | 20 |
| 189863 | 259 | 3140 | 2 |
| 189864 | 308 | 5832 | 2 |
| 189875 | 20 | 20000 | 5 |
| 189881 | 3 | 1521 | 5 |
| 189882 | 3 | 1515 | 5 |
| 189887 | 3 | 9989 | 5 |
| 189890 | 3 | 8753 | 5 |
| 189894 | 3 | 1183 | 5 |
| 189899 | 40 | 750 | 8 |
| 189902 | 3 | 10130 | 5 |
| 211720 | 220 | 9144 | 8 |
| 211721 | 13 | 5278 | 2 |