CMA-ES for Post Hoc Ensembling in AutoML: A Great Success and Salvageable Failure

Anonymous[1]

[1]Anonymous Institution

Abstract Many state-of-the-art automated machine learning (AutoML) systems use greedy ensemble selection (GES) by Caruana et al. (2004) to ensemble models found during model selection post hoc. Thereby, boosting predictive performance and likely following Auto-Sklearn 1's insight that alternatives, like stacking or gradient-free numerical optimization, overfit. Overfitting in Auto-Sklearn 1 is much more likely than in other AutoML systems because it uses only low-quality validation data for post hoc ensembling. Therefore, we were motivated to analyse whether Auto-Sklearn 1's insight holds true for systems with higher-quality validation data. Consequently, we compared the performance of covariance matrix adaptation evolution strategy (CMA-ES), state-of-the-art gradient-free numerical optimization, to GES on the 71 classification datasets from the AutoML benchmark for AutoGluon. We found that Auto-Sklearn's insight depends on the chosen metric. For the metric ROC AUC, CMA-ES overfits drastically and is outperformed by GES – statistically significantly for multi-class classification. For the metric balanced accuracy, CMA-ES does not overfit and outperforms GES significantly. Motivated by the successful application of CMA-ES for balanced accuracy, we explored methods to stop CMA-ES from overfitting for ROC AUC. We propose a method to normalize the weights produced by CMA-ES, inspired by GES, that avoids overfitting for CMA-ES and makes CMA-ES perform better than or similar to GES for ROC AUC.

Auto-Sklearn (Feurer et al., 2015) was the first automated machine learning (AutoML) system to discover that building an ensemble of models found during model selection is possible in an efficient manner and superior in predictive performance to the single best model. Afterwards, several other AutoML systems also build an ensemble *post hoc*: AutoGluon (Erickson et al., 2020), Auto-Pytorch (Mendoza et al., 2018; Zimmer et al., 2021), MLJAR (Płońska and Płoński, 2021), and H2O AutoML (LeDell and Poirier, 2020) all implemented *post hoc ensembling*.

Besides H2O AutoML, all of these systems implemented *greedy ensemble selection* (GES) (Caruana et al., 2004, 2006), a greedy search for a weight vector to aggregate the predictions of base models. In AutoML systems, GES is trained using the base models' predictions on the *validation data*, which are computed while evaluating a base model during model selection. The frequent usage of GES likely follows Auto-Sklearn's reported insight that alternatives like *stacking* (Wolpert, 1992) or gradient-free numerical optimization overfit and are more costly than GES.

Auto-Sklearn 1, by default, only has limited validation data for post hoc ensembling, that is, a 33% hold-out split of the training data. We deem this to be low-quality validation data because, depending on the dataset, 33% are not enough instances to avoid overfitting while training GES. Hence, we were motivated to analyze if Auto-Sklearn's insight also holds true for an AutoML system with higher-quality validation data, *e.g.*, AutoGluon with n-repeated k-fold cross-validation. Moreover, we were motivated to focus on gradient-free numerical optimization instead of stacking. Stacking is generally well-known in ensembling for machine learning and is used by H2O AutoML for post hoc ensembling. In contrast, gradient-free numerical optimization has not been used so far. Thus, we compare the performance of GES to *covariance matrix adaptation evolution strategy* (CMA-ES) (Hansen and Auger, 2014; Hansen, 2016), state-of-the-art gradient-free numerical optimization.

Submitted to AutoML 2023

In this study, we aim to boost the predictive performance as much as possible with post hoc ensembling. Note that GES selects a small ensemble, while methods like gradient-free numerical optimization or stacking produce an ensemble that includes all base models. Thus, the inference time and size of the final model are larger for the latter two than for GES.

Our first contribution is an application of CMA-ES for AutoGluon on the 71 classification datasets from the AutoML Benchmark (Gijsbers et al., 2022). Thereby, we show that Auto-Sklearn's insight w.r.t. overfitting of gradient-free numerical optimization depends on the chosen metric. We contradict the insight for the metric *balanced accuracy* by showing that CMA-ES statistically significantly outperforms GES. And we confirm the insight for the metric *ROC AUC* by showing that GES outperforms CMA-ES due to overfitting.

As a follow-up, our second contribution is a method to avoid overfitting for CMA-ES. Motivated by the successful application of CMA-ES for balanced accuracy, we explored methods to stop CMA-ES from overfitting to *salvage* CMA-ES for ROC AUC. We identified the chosen method to normalize the ensemble's prediction probabilities as the key to avoiding overfitting. With this knowledge, we propose a novel normalization method, inspired by GES's implicit constraints during optimization, that makes CMA-ES perform as well as GES and avoids overfitting for ROC AUC. Interestingly, our normalization method also enables us to keep the size of the ensemble small.

Our code is available on GitHub: `https://anon-github.automl.cc/r/D3C4`.

1 Related Work

Besides Auto-Sklearn 1's (Feurer et al., 2015) statement related to post hoc ensembling, only H2O AutoML names theoretical guarantees (van der Laan et al., 2007) as the reason for using stacking, but does not comment on GES. In general, details about post hoc ensembling in publications about AutoML systems were only a short comment without experiments or a reference to Auto-Sklearn 1 (Feurer et al., 2015; Mendoza et al., 2018; Erickson et al., 2020; LeDell and Poirier, 2020). We are only aware of the work by Purucker and Beel (2022), which proposed a first benchmark and framework for post hoc ensembling. The results in their Appendix also showed that GES can outperform stacking. To the best of our knowledge, no other work on post hoc ensembling for AutoML exists.

CMA-ES was previously applied to machine learning problems like hyperparameter optimization (Nomura et al., 2021; Loshchilov and Hutter, 2016) or feature weighting (Tasci et al., 2018)[1] . However, we found no work that used CMA-ES to directly optimizes the weights of an ensemble. Likewise, we have found no work that applies normalization to the solutions produced by CMA-ES nor comparable machine learning methods that apply normalization in this way to combat overfitting.

2 Application of CMA-ES for Post Hoc Ensembling

In our application of CMA-ES for post hoc ensembling, we search for an optimal weight vector $W = (w_1, ..., w_m)$ to aggregate pool P of m base models that minimizes a user-defined loss $L(P, W)$. Thereby, L aggregates the predictions of models in P by taking the W-weighted arithmetic mean.

Hence, we employ CMA-ES, as implemented in pycma (Hansen et al., 2019), with default values to find W by minimizing L. Following GES's first iteration, we set the initial solution x_0 to be the weight vector representing the single best model, that is, the weight for the single best model is one while all other models are weighted zero. The initial standard deviation is 0.2 following the intuition that a good weight vector might be close to the initial solution and that the granularity of weights can be small, e.g., between 0 and 1, like in GES.

2.1 Experiments: CMA-ES *vs.* GES

We compared CMA-ES to GES w.r.t. ROC AUC following the AutoML Benchmark (Gijsbers et al., 2022). ROC AUC requires prediction probabilities and is independent of a decision threshold that

[1] To the best of our knowledge, this work is not available in English. We read a machine-translated version.

would transform prediction probabilities into labels. We use macro average one-vs-rest ROC AUC for multiclass. We complemented the comparison by also evaluating w.r.t. balanced accuracy, which requires predicted labels and, thus, depends on a decision threshold.

For a threshold-dependent metric, the prediction of CMA-ES is, in our application, the class with the highest value after aggregating the prediction probabilities with the W-weighted mean. For a threshold-independent metric, we transform the aggregated probabilities for each instance using the softmax function, *i.e.*, we treat the aggregated probabilities of each class as decision functions and take their softmax. Otherwise, the aggregated probabilities would not represent prediction probabilities, as W can have negative or positive values of any granularity.

To compare the ensembling methods, we obtained base models and their validation data with AutoGluon (Erickson et al., 2020) for each fold of the 71 classification datasets from the AutoML benchmark (AMLB) (Gijsbers et al., 2022) – for both metrics. Then, per fold, we trained the ensemble methods on the validation data, i.e., search for W, and scored them on validation and test. The final validation/test score of a method for a dataset is the average over the 10 folds.

Following the AMLB, we ran AutoGluon for 4 hours with 8 cores (AMD EPYC 7452 CPU) and 32 GB of memory. We increased the memory for several datasets to 64 or 128 GB to avoid that insufficient memory made it impossible to produce multiple base models. In the end, AutoGluon produced between 2 and 24 base models, see Appendix D for details per dataset and metric.

We used the same resources and hardware to train and evaluate the ensemble methods. However, instead of training ensemble methods for 4 hours, we follow Auto-Sklearn's default and stop training GES after 50 iterations. This results in $m*50$ total evaluations of L by GES. Therefore, we terminated CMA-ES after $m * 50$ evaluations of L.

We included the single best base model (SingleBest) in the comparison as a baseline. To evaluate the statistical difference between the methods, we perform a Friedman test with a Nemenyi post hoc test ($\alpha = 0.05$), following the AMLB. The Friedman tests were significant in all our experiments.

2.2 Results: CMA-ES *vs.* GES

Figure 1 shows the mean rank and results of the statistical test with critical difference (CD) plots. We split the results for binary and multi-class classification in all our evaluations following the AutoML Benchmark (Gijsbers et al., 2022). We observe that CMA-ES is statistically significantly better than GES for balanced accuracy but fails to perform similarly well for ROC AUC.

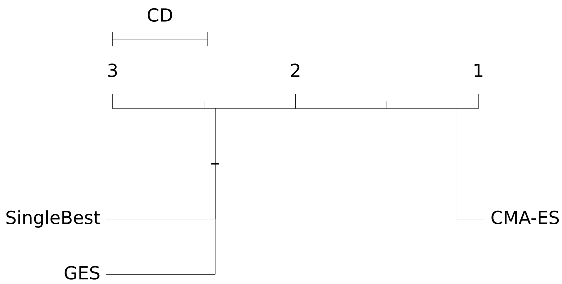
To analyze the impact of overfitting on this outcome, we inspect the change of the mean rank when switching from validation to test data for both metrics, see Table 1. While the single best is always ranked last, GES overtakes CMA-ES when switching from validation to test data for ROC AUC. Notably, CMA-ES has a mean rank of almost 1 for validation data in 3 out of 4 cases.

On validation data, GES is only competitive for multi-class ROC AUC, where it has a mean rank of 1.6. Nevertheless, GES has a larger distance to the single best on validation for balanced accuracy than it has for test data with a mean rank of $\sim$2 against the single best's $\sim$3.

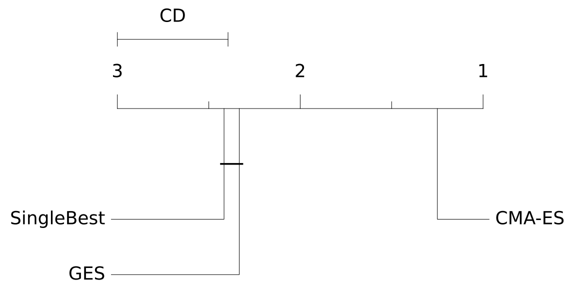
In summary, we conclude that Auto-Sklearn's insight w.r.t. overfitting does not generalize to an AutoML system with higher-quality validation data, *i.e.*, AutoGluon, for *balanced accuracy*. In contrast, *the insight holds for ROC AUC*. Furthermore, we observe that CMA-ES is able to achieve peak performance for ROC AUC on validation data.

3 Normalization to Combat Overfitting

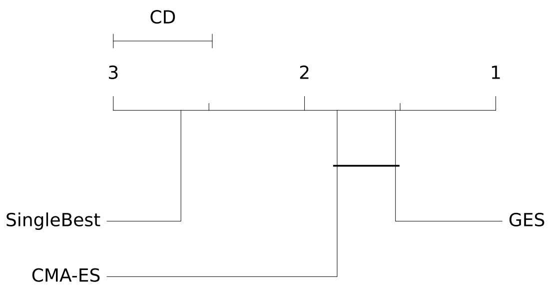
The results we just presented motivated us to salvage CMA-ES for ROC AUC. The following section describes why and how we use normalization to combat overfitting for a threshold-independent metric like ROC AUC. Since our normalization approach is inspired by GES, we start with preliminaries regarding GES and its properties.



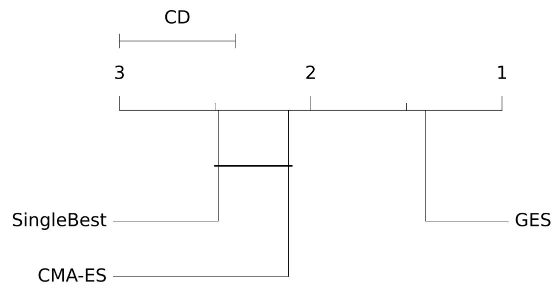
(a) Balanced Accuracy - Binary (41 Datasets)



(b) Balanced Accuracy - Multi-class (30 Datasets)



(c) ROC AUC - Binary (41)



(d) ROC AUC - Multi-class (30)

Figure 1: **CD Plots Comparing GES and CMA-ES:** The mean rank (lower is better) of a method is its line's position on the axis. Methods connected by a bar are not significantly different.

3.1 Preliminaries

Greedy ensemble selection with replacement (Caruana et al., 2004, 2006) performs an iterative greedy search to build a list of (repeated) base models, the ensemble E, that minimizes a user-defined loss function. In each iteration, the base model minimizing the loss, when added to E, is *selected* to be part of E. To produce predictions and evaluate any E, the (repeated) predictions of all base models in E are aggregated with the arithmetic mean. Taking the arithmetic mean of E weights base models that exit multiple times higher. Hence, given E, we can compute a weight vector. Assuming we run GES for N iterations[2], then $|E| = N$ and we compute the weight vector using:

$$W^{pDisc} = \left[\frac{countIn(p_i, E)}{N} \;\middle|\; p_i \in P \right]. \tag{1}$$

While analysing GES, we found two constraints of the weight vector W^{pDisc} that we believe to be essential for its performance. That is, W^{pDisc} is *pseudo-discrete* and *sparse*. Both properties are only *implicitly* respected by GES and were, to the best of our knowledge, never formally defined.

[2]We always denote N as the number of the iteration the final E was found in. Depending on the implementation of GES, the final E does not need to be from the final iteration.

Table 1: Mean rank change from validation to test data for CMA-ES compared to GES and SingleBest.

Metric	Task Type	Mean $Rank_{Validation}$	Mean $Rank_{Test}$	Absolute Rank (Val $\rightarrow$ Test)
Balanced Accuracy	Binary	1.00	1.12	1.0 $\rightarrow$ 1.0
Balanced Accuracy	Multi-class	1.03	1.25	1.0 $\rightarrow$ 1.0
ROC AUC	Binary	1.02	1.83	1.0 $\rightarrow$ 2.0
ROC AUC	Multi-class	1.42	2.12	1.0 $\rightarrow$ 2.0

Pseudo-Discrete. We call W^{pDisc} pseudo-discrete because one can transform every weight vector produced by GES into a discrete count of how often a base model has been selected. This can be done by multiplying W^{pDisc} with N, reversing Equation 1. In fact, every weight vector produced by GES is in the set $\mathcal{G} = \{W' \mid W' \in H(N) \text{ and } \sum_{i=1}^{m} w_i = 1\}$ with $H(N)$ the m-fold Cartesian product of $\{0, 1/N, 2/N, ..., 1\}$:

$$H(N) = \{0, 1/N, 2/N, \ldots, 1\} \times \cdots \times \{0, 1/N, 2/N, \ldots, 1\}. \tag{2}$$

In other words, every weight $w_i \in W^{pDisc}$ can be expressed as a positive fraction with denominator N, and the weight vector sums to 1. This follows from GES iteratively building a list of base models E and calculating the final weight vector with Equation 1.

We would like to remark that this formulation of GES is very similar to mallows' model average (MMA) (Hansen, 2007, 2008; Le and Clarke, 2022) and that GES might share MMA's asymptotic guarantees for regression if L is the squared error (Le and Clarke, 2022).

Sparse. W^{pDisc} is sparse, that is, a weight vector where many models are assigned zero weight – as intended for an *ensemble selection* approach (Tsoumakas et al., 2009). To the best of our knowledge, a guarantee for sparseness was never formally introduced or proven for (greedy) ensemble selection, cf. (Caruana et al., 2004, 2006; Tsoumakas et al., 2009). Here, we shortly provide an argument for why it is likely that GES produces a sparse weight vector:

GES only adds new base models to E if they reduce the loss. Hence, it would require at least m iterations where adding a new base model would reduce the loss more than adding an existing base model again (increasing its weight). As a result, for appropriate values for m and N, it is unlikely that enough iterations happened such that each model was able to reduce the loss once. Auto-Sklearn, for example, uses $m = 50$ and $N = 50$ by default. Moreover, once E becomes large, the changes to the aggregated prediction that are induced by adding a new base model are minimal. Thus, it also becomes less likely that the changes result in a different loss. Additonally, the larger E is, the more likely GES has reached a (local) optimum, which can not be improved upon by adding new models. In short, the iterative greedy approach to add models to E likely makes W^{pDisc} sparse.

3.2 Motivation

Since all solutions produced by GES are pseudo-discrete and (likely) sparse, and since GES does not seem to overfit, we hypothesized that both properties might help to avoid overfitting.

Note, the properties can be seen as constraints. They constrain the weight vector to be sparse, sum to 1, and contain only values such that $0 \leq w_i \leq 1$. In contrast, our application of CMA-ES uses no such constraints. By default, CMA-ES produces a continuous and dense vector which does not need to sum to 1 and may contain negative or positive values of any granularity.

Thus, our first idea was to constrain the optimization process of CMA-ES such that it would produce results that match the constraints of GES. However, we found that once the same constraints are introduced, CMA-ES often violates the constraints; making CMA-ES inefficient and often leading to an endless loop due to rejection sampling. In other words, we were not able to make CMA-ES produces solution vectors that fulfill all constraints of GES. In general, constraining CMA-ES is also not trivial (Biedrzycki, 2020), and we leave more sophisticated approaches to constrain CMA-ES for post hoc ensembling, like methods based on repair-and-inject or penalization (Hansen, 2016) or with relaxed constraints, to future work.

Instead of constraining the optimization process of CMA-ES, we moved to adding the constraints directly to the weight vector when they are evaluated, following a concept observed from GES. That is, we observed that while the constraints of GES are an implicit result of the algorithm as defined by Caruana et al. (2004), they manifested explicitly only when one computes the weight vector with Equation 1. The optimization loop of GES, *i.e.*, iteratively building E, does not explicitly consider these constraints, but only greedily minimizes a user-defined loss. In other words, the optimizer is only implicitly constrained *by applying constraints during the computation of the weight vector; before evaluating the vector's performance.*

In detail, every time GES computes the loss for an ensemble E, it first transforms E into W^{pDisc} using Equation 1. Thereby, applying the constraints that the resulting weight vector must sum to 1, is sparse, and $0 \leq w_i \leq 1$. Then, the $L(P, W^{pDisc})$ is returned as the loss of E. At this point, it becomes clear that changing Equation 1 leads to different constraints; the loss of E could change without touching the optimization loop of GES.

As a result, we were motivated to apply the same concept to CMA-ES by normalizing the weight vector before we aggregate the predictions of the base models. Thus, changing the loss associated with a weight vector proposed by CMA-ES outside of its optimization process. In contrast, our application in Section 2 normalized the aggregated predictions for ROC AUC using softmax – we normalized *after aggregation*. Now, however, we propose to normalize *before aggregation* as in GES. In turn, this also changes the optimization process of CMA-ES, *e.g.*, the parameter update, because a weight vector might have a different loss depending on normalizing before or after aggregation.

3.3 Normalization Methods

We propose three distinct normalization methods. Two of the methods we propose are based on the concept of GES such that the last proposed method tries to simulate Equation 1 fully.

1) Softmax (CMA-ES-Softmax). Initially, we propose a simple alternative to our previous usage of CMA-ES by moving the (non-linear) softmax before the aggregation. That is, we normalize the weight vector by taking its softmax, resulting in W^s with $\sum_{j=1}^{m} w_j^s = 1$ and $0 \leq w_j \leq 1$ for $w_j \in W^s$.

2) Softmax & Implict GES Normalization (CMA-ES-ImplictGES). Next, we propose to re-normalize W^s with the aim of producing an equivalent to a pseudo-discrete weight vector W^{pDisc}; simulating GES's $\mathcal{G}$ (see Equation 2). Therefore, we round each value of W^s to the nearest fraction with denominator N_{hyp} producing a *rounding-discrete* weight vector W^{rDisc}. Then, N_{hyp} represents the number of *hypothetical iterations* for a simulated $\mathcal{G}$. We set $N_{hyp} = 50$, similar to GES.

We produce W^{rDisc} by multiplying W^s with N_{hyp} and rounding each element to the nearest integer afterwards; rounding up for values larger than 0.5. Note, that the resulting integer vector can be thought of as a vector of repetitions $R = (r_1, ..., r_m)$ where r_i denotes how often a model has been repeated in a hypothetical list of repeated base model E_{hyp}. That is, E_{hyp} is connected to W^{rDisc} like an E to its W^{pDisc}. Hence, we can compute W^{rDisc} using R, paralleling Equation 1:

$$W^{rDisc} = \left[\frac{r_i}{\sum_{j=1}^{m} r_j} \mid r_i \in R \right]. \tag{3}$$

W^{rDisc} sums to 1, and each element is between 0 and 1. Interestingly, we found that this approach also *implicitly trims* base models, as the nearest fraction can be $\frac{0}{N_{hyp}}$ such that the method assigns zero weight to base models in these cases.

3) Softmax & Explicit GES Normalization (CMA-ES-ExplicitGES). Finally, we propose to explicitly trim base models and perfect the simulation of Equation 1. We can explicitly trim base models based on N_{hyp}. We found that a weight w_j^s is set to zero by rounding if $w_j^s * N_{hyp} \leq 0.5$. If we reformulate the inequality to $w_j^s \leq 0.5 * \frac{1}{N_{hyp}}$, we see that this parallels GES, where the number of iterations determines the minimal weight a model can be assigned, *i.e.*, $\frac{1}{N}$.

Furthermore, we found that CMA-ES-ImplictGES does not simulate GES sufficiently. We observed that rounding may result in $\sum_{j=1}^{m} r_j \neq N_{hyp}$. That is, the total number of repetitions in R did not match the number of simulated iterations nor the (hypothetical) length of E_{hyp}. R was supposed to relate to E_{hyp} for W^{rDisc} like an E to its W^{pDisc}. Yet for GES, it holds that $|E| = N$ while $|E_{hyp}| \neq N_{hyp}$ can happen in CMA-ES-ImplictGES.

Considering both, we implemented the third method as follows. First, we trim any base model smaller than $\frac{0.5}{N_{hyp}}$ by setting their weight to zero. If we set all weights to zero, we fall back to an unweighted average. Second, we round to the nearest integer, producing R'. In the following we denote $\sum_{j=1}^{m} r'_j$ as $\sum R'$. Next, we modify R' into R'_{new} to achieve $\sum R'_{new} = N_{hyp}$. We want to keep the distribution of R'_{new} as close as possible to the distribution of R'. Hence, we keep the relative distances between the individual elements in R' and R'_{new} similar.

To do so, we start with $R'_{new} = R'$. If $\sum R' > N_{hyp}$, we decrement elements in R'_{new} by 1 until $\sum R'_{new} = N_{hyp}$. We decrement in order from lowest to highest valued element in R', that is, lowest to highest weighted base model in the resulting weight vector. Thus, first trimming base models with only one repetition. Finally, if $R' - N_{hyp}$ is large enough, we decrement the most repeated elements. Note, due to rounding, we must decrement each element once in the worst case. If $\sum R' < N_{hyp}$, we have to increase the value of elements in R'_{new}. To keep the relative distances similar, we equally distributed $N_{hyp} - R'$ increments between all non-zero elements in R'_{new}.

Finally, the R'_{new} is transformed into a weight vector with Equation 3.

3.4 Comparing Normalization Methods

We use CMA-ES-ExplicitGES for the final evaluation below because it is the only approach that is in line with GES's concepts. Nevertheless, here, we shortly compare the three normalization methods on the same data as used in Section 2.1. We run CMA-ES, as described above, with the three different methods for normalization on the data from AutoGluon for ROC AUC. We ignore the threshold-dependent balanced accuracy because CMA-ES is not affected by overfitting for balanced accuracy. Besides normalization, the main difference to the application from Section 2 is that we do not apply softmax after aggregation anymore when we apply normalization.

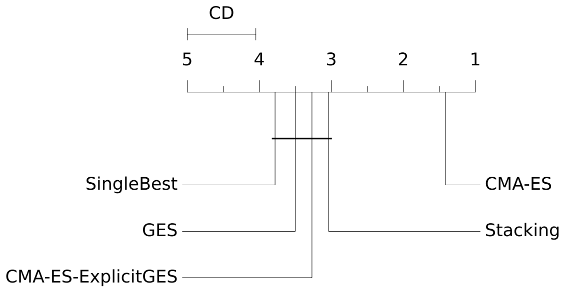
First, a note regarding sparseness. On average, across all datasets for ROC AUC, ∼13.2 base models exist. We observe that CMA-ES without normalization has an average ensemble size, that is, the number of non-zero weighted base models, of ∼12.9. In contrast, CMA-ES-ExplicitGES has an average ensemble size of ∼6.3, CMA-ES-ImplicitGES of ∼5.4. For context, GES has an average ensemble size of ∼5.8 Hence, we conclude that CMA-ES produces dense weight vectors. While our normalization approaches are able to produce sparse vectors like GES.

Next, we repeat the statistical test performed in Section 2.1 for all normalization methods, CMA-ES, and the SingleBest, see Figure 4 in the Appendix E. We observe that all normalization methods outperform CMA-ES and that CMA-ES-ExplicitGES ranks highest. Furthermore, the different normalization methods are not statistically significantly different from each other. Only CMA-ES-ExplicitGES is significantly different from CMA-ES for multi-class.

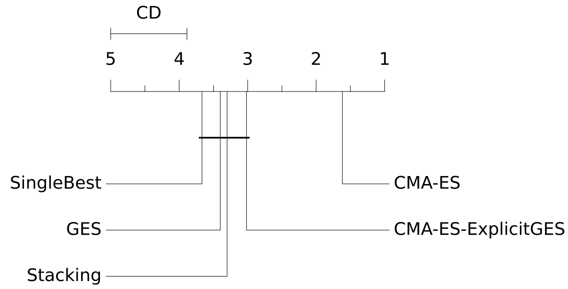
4 Overall Experiments

In our final evaluation, we mirror the experiments from Section 2.1 and compare the SingleBest, GES, stacking, CMA-ES, and CMA-ES with normalization (CMA-ES-ExplicitGES). For our implementation of stacking (Wolpert, 1992), we use a default Logistic Regression classifier from scikit-learn (Pedregosa et al., 2011) as a stacking model. We adjusted the code such that we terminate after $m * 50$ evaluations to make the method comparable to GES and CMA-ES. For CMA-ES we stick to the implementation and default hyperparameters as described in Section 2.

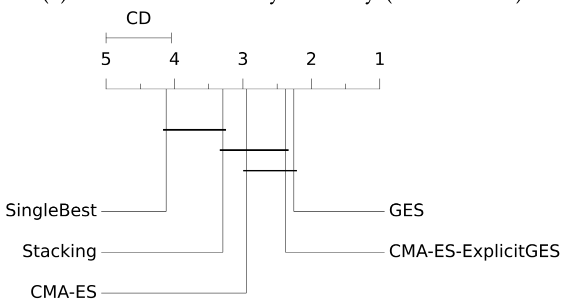
Besides the statistical tests, we also inspect the difference in the distributions of relative performance. Therefore, we follow the AutoML benchmark (Gijsbers et al., 2022) and use *normalized improvement* to make the scores of methods comparable across different datasets. We scale the scores for a dataset such that -1 is equal to the score of a baseline, here the SingleBest, and 0 is equal to the score of the best method on the dataset. We employ a variant of normalized improvement as we ran into an edge case where the normalized improvement is undefined if the difference between the single best model and the best method is 0. In our variant, for this edge case, we set everything as good as the SingleBest to -1 and penalize all methods worse than the baseline with -10.



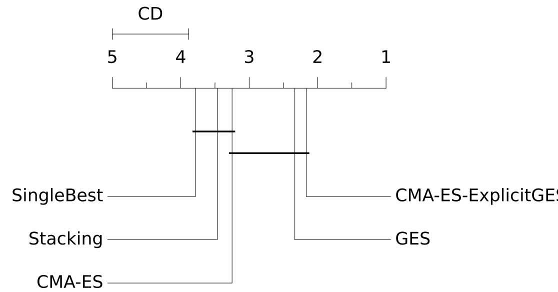
(a) Balanced Accuracy - Binary (41 Datasets)



(b) Balanced Accuracy - Multi-class (30 Datasets)



(c) ROC AUC - Binary (41)



(d) ROC AUC - Multi-class (30)

Figure 2: **CD Plots for all Methods**: Methods connected by a bar are not significantly different.

5 Overall Results

Figure 2 shows the results of the statistical tests and mean rankings for the compared methods. The distribution of the relative performance is shown in Figure 3.

Overall Predictive Performance. All post hoc ensembling methods always outperform the SingleBest on average, although not always statistically significant – see Figure 2. Yet, post hoc ensembling can overfit and become worse for specific datasets, as indicated by the black dots left of the red bar and the number of outliers in square brackets in Figure 3.

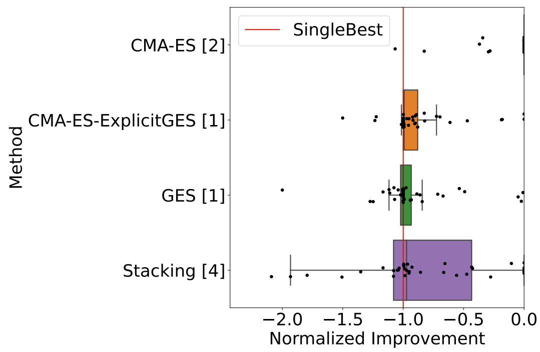
For balanced accuracy, we observe that CMA-ES significantly beats all methods. Likewise, we observe that stacking and CMA-ES-ExplicitGES outperform GES by a small non-significant margin.

For ROC AUC, we see that GES and CMA-ES-ExplicitGES outperform all other methods and differ only by a small non-significant margin. Both are also significantly different from the SingleBest; unlike stacking. Moreover, Figure 3 shows us that CMA-ES-ExplicitGES has similar or better relative performance distributions than GES (see the medians and whiskers).

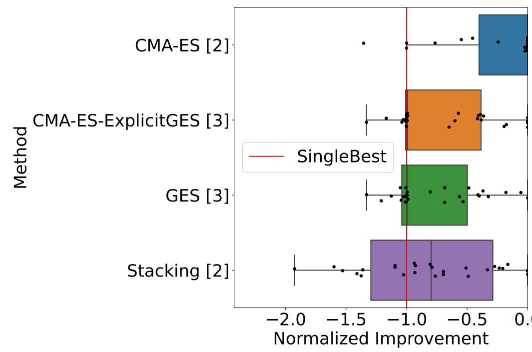
Normalization to Combat Overfitting. See Table 2 to inspect overfitting for CMA-ES-ExplictGES. In general, CMA-ES-ExplictGES's mean rank, compared to GES and the SingleBest, changes only minimally between validation and test data. Showing us that it overfits less than CMA-ES (compare to Table 1, Section 2.2). As before, the SingleBest is always the worst-ranked method. GES is worse than CMA-ES-ExplictGES on test data for all but ROC AUC Binary. On validation data, however, GES is better than CMA-ES-ExplictGES in all cases except for ROC AUC multi-class, where it is tied. Now, GES is *more affected by overfitting* than CMA-ES with normalization.

6 Conclusion

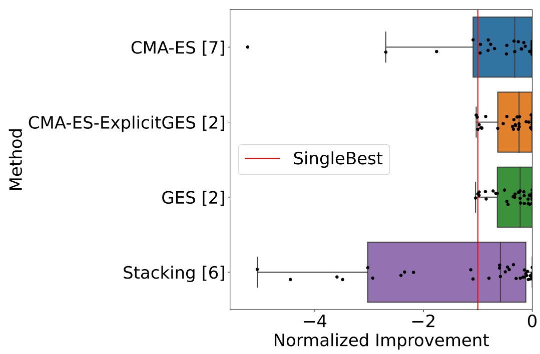
Greedy ensemble selection (GES) (Caruana et al., 2004) is often used for post hoc ensembling in AutoML; likely as a result of Auto-Sklearn 1's (Feurer et al., 2015) reported insight that GES is superior to potential alternatives, like gradient-free numerical optimization, for post hoc ensembling.



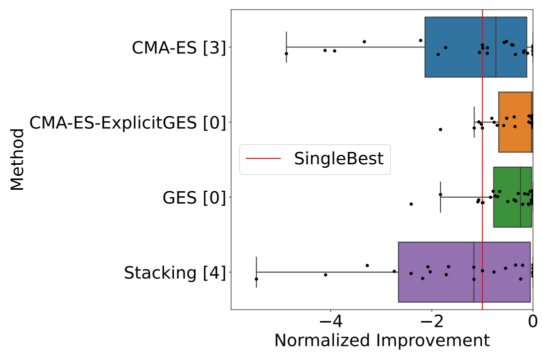
(a) Balanced Accuracy - Binary (41 Datasets)



(b) Balanced Accuracy - Multi-class (30 Datasets)



(c) ROC AUC - Binary (41)



(d) ROC AUC - Multi-class (30)

Figure 3: **Normalized Improvement Boxplots**: Higher normalized improvement is better. Each black point represents the improvement for one dataset. A value smaller than −1 is worse than the single best model (red vertical line), while 0 is the best observed value. The number in square brackets counts the outliers of a method left of the plot's boundary.

Table 2: Mean rank change for CMA-ES-ExplictGES compared to GES and SingleBest.

Metric	Task Type	Mean $Rank_{Validation}$	Mean $Rank_{Test}$	Absolute Rank (Val → Test)
Balanced Accuracy	Binary	1.74	1.78	2.0 → 1.0
Balanced Accuracy	Multi-class	1.73	1.78	2.0 → 1.0
ROC AUC	Binary	1.63	1.70	2.0 → 2.0
ROC AUC	Multi-class	1.50	1.57	1.5 → 1.0

In this paper, we have shown that Auto-Sklearn's insight w.r.t. overfitting depends on the metric when tested for an AutoML system with higher-quality validation data than Auto-Sklearn, e.g., AutoGluon (Erickson et al., 2020). Indeed, for the metric ROC AUC, GES does not overfit meaningfully, while gradient-free numerical optimization, e.g., CMA-ES (Hansen and Auger, 2014; Hansen, 2016), overfits drastically. However, for balanced accuracy, CMA-ES does not overfit and outperforms GES.

As a direct consequence, we were motivated to find a method that combats the overfitting of CMA-ES for ROC AUC. Therefore, we proposed a novel normalization method, is inspired by GES, which successfully salvages CMA-ES for ROC AUC by making CMA-ES perform better than or similar to GES.

References

Biedrzycki, R. (2020). Handling bound constraints in CMA-ES: an experimental study. *Swarm Evol. Comput.*, 52.

Caruana, R., Munson, A., and Niculescu-Mizil, A. (2006). Getting the most out of ensemble selection. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 828–833. IEEE Computer Society.

Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. (2004). Ensemble selection from libraries of models. In Brodley, C. E., editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM.

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., and Smola, A. J. (2020). Autogluon-tabular: Robust and accurate automl for structured data. *CoRR*, abs/2003.06505.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2962–2970.

Gijsbers, P., Bueno, M. L. P., Coors, S., LeDell, E., Poirier, S., Thomas, J., Bischl, B., and Vanschoren, J. (2022). AMLB: an automl benchmark. *CoRR*, abs/2207.12560.

Hansen, B. E. (2007). Least squares model averaging. *Econometrica*, 75(4):1175–1189.

Hansen, B. E. (2008). Least-squares forecast averaging. *Journal of Econometrics*, 146(2):342–350.

Hansen, N. (2016). The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772.

Hansen, N., Akimoto, Y., and Baudis, P. (2019). CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634.

Hansen, N. and Auger, A. (2014). Principled design of continuous stochastic search: From theory to practice. In Borenstein, Y. and Moraglio, A., editors, *Theory and Principled Methods for the Design of Metaheuristics*, Natural Computing Series, pages 145–180. Springer.

Le, T. M. and Clarke, B. S. (2022). Model averaging is asymptotically better than model selection for prediction. *J. Mach. Learn. Res.*, 23:33:1–33:53.

LeDell, E. and Poirier, S. (2020). H2O AutoML: Scalable automatic machine learning. *7th ICML Workshop on Automated Machine Learning (AutoML).*

Loshchilov, I. and Hutter, F. (2016). CMA-ES for hyperparameter optimization of deep neural networks. *CoRR*, abs/1604.07269.

Mendoza, H., Klein, A., Feurer, M., Springenberg, J. T., Urban, M., Burkart, M., Dippel, M., Lindauer, M., and Hutter, F. (2018). Towards automatically-tuned deep neural networks. In Hutter, F., Kotthoff, L., and Vanschoren, J., editors, *AutoML: Methods, Sytems, Challenges*, chapter 7, pages 141–156. Springer.

Nomura, M., Watanabe, S., Akimoto, Y., Ozaki, Y., and Onishi, M. (2021). Warm starting CMA-ES for hyperparameter optimization. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 9188–9196. AAAI Press.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Płońska, A. and Płoński, P. (2021). Mljar: State-of-the-art automated machine learning framework for tabular data. version 0.10.3.

Purucker, L. and Beel, J. (2022). Assembled-openML: Creating efficient benchmarks for ensembles in autoML with openML. In *First Conference on Automated Machine Learning (Late-Breaking Workshop)*.

Tasci, E., Gökalp, O., and Ugur, A. (2018). Development of a novel feature weighting method using CMA-ES optimization. In *26th Signal Processing and Communications Applications Conference, SIU 2018, Izmir, Turkey, May 2-5, 2018*, pages 1–4. IEEE.

Tsoumakas, G., Partalas, I., and Vlahavas, I. P. (2009). An ensemble pruning primer. In Okun, O. and Valentini, G., editors, *Applications of Supervised and Unsupervised Ensemble Methods*, volume 245 of *Studies in Computational Intelligence*, pages 1–13. Springer.

van der Laan, M. J., Polley, E. C., and Hubbard, A. E. (2007). Super learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1).

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2):241–259.

Zimmer, L., Lindauer, M., and Hutter, F. (2021). Auto-pytorch tabular: Multi-fidelity metalearning for efficient and robust autodl. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 3079 – 3090. also available under https://arxiv.org/abs/2006.13799.

A Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We state in the abstract and introduction that we compare GES to CMA-ES w.r.t. overfitting. Moreover, we claim to look at normalization to avoid overfitting. This is exactly what we do in the paper.

   (b) Did you describe the limitations of your work? [Yes] In the Appendix, see B.

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In the Appendix, see C.

   (d) Have you read the ethics author's and review guidelines and ensured that your paper conforms to them? `https://2023.automl.cc/ethics/` [Yes] We believe our paper confirms to them.

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A] We included no theoretical results; only theoretical arguments for our proposed normalization method.

   (b) Did you include complete proofs of all theoretical results? [N/A] We included no theoretical results; only theoretical arguments for our proposed normalization method.

3. If you ran experiments...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit version), an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] See our code repository (mentioned in the introduction) for all details.

   (b) Did you include the raw results of running the given instructions on the given code and data? [Yes] See our code repository.

   (c) Did you include scripts and commands that can be used to generate the figures and tables in your paper based on the raw results of the code, data, and instructions given? [Yes] See our code repository.

   (d) Did you ensure sufficient code quality such that your code can be safely executed and the code is properly documented? [Yes] We believe that our code quality and documentation are sufficient.

   (e) Did you specify all the training details (e.g., data splits, pre-processing, search spaces, fixed hyperparameter settings, and how they were chosen)? [Yes] See the Section 2.1 and 4. Additionally, see our code repository.

   (f) Did you ensure that you compared different methods (including your own) exactly on the same benchmarks, including the same datasets, search space, code for training and hyperparameters for that code? [Yes] We ran all methods on the same data.

   (g) Did you run ablation studies to assess the impact of different components of your approach? [Yes] We compared different normalization approaches, see Section 3.4.

   (h) Did you use the same evaluation protocol for the methods being compared? [Yes] We ran all methods on the same data with the same evaluation protocol and code.

(i) Did you compare performance over time? [No] We compared performance for a specific point in time (after 50 iterations of GES, i.e., after $m * 50$ function evaluations of L). Performance over time was out of scope for our experiments.

(j) Did you perform multiple runs of your experiments and report random seeds? [Yes] Yes, we used 10-fold cross-validation for all our runs. The used random seeds can be found in our code.

(k) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We took the average over the 10 folds as a score following previous work and have not reported variance across folds.

(l) Did you use tabular or surrogate benchmarks for in-depth evaluations? [N/A] Such benchmarks were not available for our use case.

(m) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 2.1.

(n) Did you report how you tuned hyperparameters, and what time and resources this required (if they were not automatically tuned by your AutoML method, e.g. in a NAS approach; and also hyperparameters of your own method)? [N/A] We did not tune hyperparameters. We used a default application of CMA-ES and introduced no meaningful new hyperparameters with our approaches that would require tuning.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] See Section 2.1 and Appendix F.

(b) Did you mention the license of the assets? [Yes] See Appendix F.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See our code repository.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We are only using publicly available data that was used before in benchmarks.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We believe that the data we are using does not contain personally identifiable information or offensive content.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We did not use crowdsourcing or conducted research with human subjects.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] We did not use crowdsourcing or conducted research with human subjects.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We did not use crowdsourcing or conducted research with human subjects.

B Limitations

We note that our work is limited with respect to the following points: 1) we did not explore variations (w.r.t. hyperparameters or implementation) of CMA-ES in our work; 2) we considered overfitting with respect to mean rank change between validation and test data, but did not consider other concepts of overfitting; 3) we only looked at normalization to combat overfitting for CMA-ES and were not able to compare normalization to using constraints during optimization; and 4) we only provided a high-level theoretical analysis of GES and were not able to provide more fundamental work or proofs.

C Broader Impact Statement

After careful reflection, we determine that this work presents *almost* no notable or new negative impacts to society or the environment that are not already present for existing state-of-the-art AutoML systems. This follows from our work being mostly domain-independent, abstract, and methodical. We only proposed to replace one component of an AutoML system such that the predictive performance improves. Nevertheless, we would like to remark that our work might prompt others to use a default application of CMA-ES instead of GES for a metric like balanced accuracy. This might have a negative impact on the environment because this would likely increase the inference time and size of the final ensemble proposed by AutoML systems.

In contrast – as a trade-off – we see the positive impact that higher predictive performance with CMA-ES could better support decisions made with AutoML systems. Moreover, we believe that our work might help to understand GES, the currently most used method, better; such that its performance and behaviour becomes more explainable.

D Data Overview

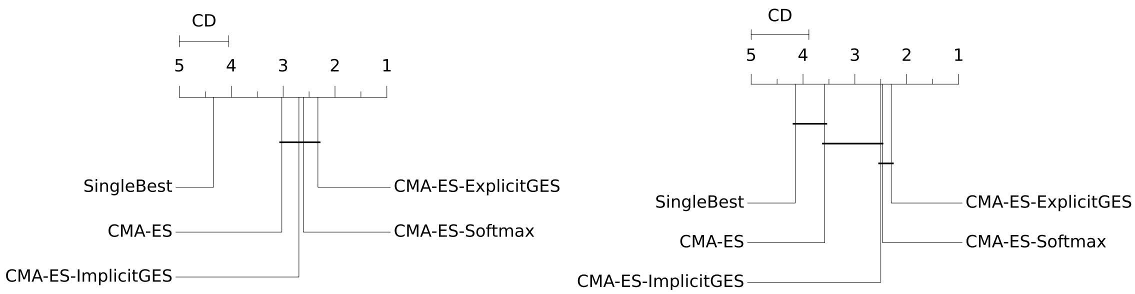
See Table 3 for an overview of the used datasets and their characteristics. Additionally, the table shows the mean number of base models and the mean number of distinct algorithms generated by AutoGluon for the dataset for each metric (mean over the 10 folds of a dataset).

Table 3: Data Overview

						Mean # Base Models		Mean # Distinct Algorithms	
Dataset Name	OpenML Task ID	#instances	#features	#classes	Memory (GB)	Balanced Accuracy	ROC AUC	Balanced Accuracy	ROC AUC
yeast	2073	1484	9	10	32	21.0	21.3	12.0	12.1
KDDCup09_appetency	3945	50000	231	2	32	11.0	11.0	11.0	11.0
covertype	7593	581012	55	7	64	13.3	12.9	8.3	8.0
amazon-commerce-reviews	10090	1500	10001	50	32	8.3	8.7	5.3	5.7
Australian	146818	690	15	2	32	13.0	13.0	13.0	13.0
wilt	146820	4839	6	2	32	12.0	12.0	12.0	12.0
numerai28.6	167120	96320	22	2	32	12.0	12.0	12.0	12.0
phoneme	168350	5404	6	2	32	12.0	11.9	12.0	11.9
credit-g	168757	1000	21	2	32	13.0	13.0	13.0	13.0
steel-plates-fault	168784	1941	28	7	32	21.0	21.0	12.0	12.0
APSFailure	168868	76000	171	2	32	12.0	12.0	12.0	12.0
dilbert	168909	10000	2001	5	32	12.9	12.7	7.9	8.3
fabert	168910	8237	801	7	32	19.8	19.8	11.0	11.0
jasmine	168911	2984	145	2	32	12.3	12.9	12.3	12.9
airlines	189354	539383	8	2	64	9.0	8.9	9.0	8.9
dionis	189355	416188	61	355	128	4.0	4.2	3.0	3.6
albert	189356	425240	79	2	64	7.0	7.0	7.0	7.0
gina	189922	3153	971	2	32	12.0	12.0	12.0	12.0
ozone-level-8hr	190137	2534	73	2	32	13.0	13.0	13.0	13.0
vehicle	190146	846	19	4	32	24.0	24.0	13.0	13.0
madeline	190392	3140	260	2	32	12.0	12.3	12.0	12.3
philippine	190410	5832	309	2	32	12.0	12.0	12.0	12.0
ada	190411	4147	49	2	32	12.0	12.0	12.0	12.0
arcene	190412	100	10001	2	32	13.0	13.0	13.0	13.0
jannis	211979	83733	55	4	32	14.9	15.3	9.2	9.3
Diabetes130US	211986	101766	50	3	32	17.9	18.1	10.8	10.9
micro-mass	359953	571	1301	20	32	13.0	13.0	13.0	13.0
eucalyptus	359954	736	20	5	32	13.0	13.0	13.0	13.0
blood-transfusion-service-center	359955	748	5	2	32	13.0	13.0	13.0	13.0
qsar-biodeg	359956	1055	42	2	32	13.0	13.0	13.0	13.0
cnae-9	359957	1080	857	9	32	20.0	20.0	11.0	11.0
pc4	359958	1458	38	2	32	13.0	13.0	13.0	13.0
cmc	359959	1473	10	3	32	21.8	21.6	12.0	12.0
car	359960	1728	7	4	32	18.4	18.0	9.2	9.0
mfeat-factors	359961	2000	217	10	32	20.0	20.0	11.0	11.0
kc1	359962	2109	22	2	32	12.8	13.0	12.8	13.0
segment	359963	2310	20	7	32	21.0	21.0	12.0	12.0
dna	359964	3186	181	3	32	19.0	19.0	10.0	10.0
kr-vs-kp	359965	3196	37	2	32	10.0	10.0	10.0	10.0
Internet-Advertisements	359966	3279	1559	2	32	12.0	12.0	12.0	12.0
Bioresponse	359967	3751	1777	2	32	12.0	12.0	12.0	12.0
churn	359968	5000	21	2	32	12.0	12.0	12.0	12.0
first-order-theorem-proving	359969	6118	52	6	32	20.1	20.1	11.1	11.1
GesturePhaseSegmentationProcessed	359970	9873	33	5	32	20.0	20.4	11.0	11.2
PhishingWebsites	359971	11055	31	2	32	10.0	10.0	10.0	10.0
sylvine	359972	5124	21	2	32	12.0	12.0	12.0	12.0
christine	359973	5418	1637	2	32	12.0	12.0	12.0	12.0
wine-quality-white	359974	4898	12	7	32	21.0	21.0	12.0	12.0
Satellite	359975	5100	37	2	32	12.0	12.0	12.0	12.0
Fashion-MNIST	359976	70000	785	10	64	12.1	13.0	8.5	8.2
connect-4	359977	67557	43	3	32	16.2	16.3	9.0	9.0
Amazon_employee_access	359979	32769	10	2	32	9.1	10.0	9.1	10.0
nomao	359980	34465	119	2	32	12.0	10.0	12.0	10.0
jungle_chess_2pcs_raw_endgame_complete	359981	44819	7	3	32	19.5	19.9	11.0	11.0
bank-marketing	359982	45211	17	2	32	12.0	12.0	12.0	12.0
adult	359983	48842	15	2	32	12.0	11.9	12.0	11.9
helena	359984	65196	28	100	32	7.7	7.9	5.0	5.0
volkert	359985	58310	181	10	32	13.9	12.5	8.9	8.6
robert	359986	10000	7201	10	64	9.7	9.3	7.6	7.3
shuttle	359987	58000	10	7	32	18.9	19.0	11.0	11.0
guillermo	359988	20000	4297	2	32	9.0	9.0	9.0	9.0
riccardo	359989	20000	4297	2	32	10.1	9.0	10.1	9.0
MiniBooNE	359990	130064	51	2	32	10.3	10.2	10.3	10.2
kick	359991	72983	33	2	32	11.7	12.0	11.7	12.0
Click_prediction_small	359992	39948	12	2	32	11.8	12.4	11.8	12.4
okcupid-stem	359993	50789	20	3	32	18.6	19.8	11.0	11.4
sf-police-incidents	359994	2215023	9	2	64	11.0	7.2	11.0	7.2
KDDCup99	360112	4898431	42	23	128	10.3	7.6	8.5	6.9
porto-seguro	360113	595212	58	2	64	6.2	2.2	6.2	2.2
Higgs	360114	1000000	29	2	64	3.0	3.0	3.0	3.0
KDDCup09-Upselling	360975	50000	14892	2	128	10.5	9.0	10.5	9.0

E Comparison of Normalization Methods

See Figure 4 for a comparison of the three proposed normalization methods following the experiments described in Section 2.1.



(a) ROC AUC - Binary

(b) ROC AUC - Multi-class

Figure 4: **CD Plots Comparing the Normalization Methods for ROC AUC**: Mean rank of the methods (lower is better). Methods connected by a bar are not significantly different.

F Used Assets: Essential Python Frameworks for the Implementation and Experiments

The following frameworks were essential for our implementation and experiments:

- AutoGluon (Erickson et al., 2020), Version: 0.6.2, Apache-2.0 License; We used AutoGluon to generate base models for post hoc ensembling.
- pycma (Hansen et al., 2019), Version 3.2.2, BSD 3-Clause License; We used pycma for CMA-ES.
- Assembled (Purucker and Beel, 2022), Version 0.0.4, MIT License; We used Assembled to store the base models generated with AutoGluon and to run our ensemble-related experiments.