Adaptivee: Adaptive Ensemble for Tabular Data

Anonymous Author(s)
Affiliation
Address
email

Abstract

Ensemble methods are widely used to improve model performance by combining multiple models, each contributing uniquely to predictions. Traditional ensemble approaches often rely on static weighting schemes that do not account for the varying effectiveness of individual models across different subspaces of the data. This work introduces *adaptivee*, a dynamic ensemble framework designed to optimize performance for tabular data tasks by adjusting model weights in response to specific data characteristics. The *adaptivee* framework offers flexibility through various reweighting strategies, including emphasizing single models for subspace specialization or distributing importance among models for robustness. Experiments on the OpenML-CC18 benchmark demonstrate that *adaptivee* can significantly boost performance, achieving up to a 0.6% improvement in balanced accuracy over AutoGluon ensembling strategies. This framework opens new avenues for advancing ensemble techniques, particularly in tabular data contexts where model complexity is constrained by the nature of the data.

1 Introduction

In the tabular data realm, using ensemble models, i.e. combination of multiple models, to outweigh the performance of each of them separately is a well-known practice [Zhou, 2012; Singh et al., 2007; Cabrera et al., 2008]. While this technique is mostly used in established tree-based models like random forest [Breiman, 2001] or AdaBoost [Freund and Schapire, 1997], it is also beneficial to apply it to the set of more complex models of different types to leverage different ways of capturing relations between data [Sesmero et al., 2015]. In this scenario, it is crucial to ensure that the best model in an ensemble has the greatest impact on the final prediction. This is done by assigning to each model the weights corresponding to their performance metric, such as accuracy [Erickson et al., 2020].

This approach, along with the methods to obtain the weights mentioned above, was extensively researched in recent years [Rokach, 2019]. While being a universal technique that boosts final performance in tabular tasks, it misses important aspects of using multiple types of models at once: different models capture different aspects of data and thus, perform better on different subspaces of the data. Hence, a question arises: can one adjust model weighting in the ensemble to let the better model on a certain subspace of data have the highest impact on prediction, regardless of the subspace? While universal for all types of supervised machine learning, this query is especially important for tabular tasks, where one cannot improve their model by simply increasing in complexity due to data specification.

In this paper, we introduce the Adaptive Ensemble (*adaptivee*), a framework to address the problem of a dynamic reweighting ensemble to gain additional boost in the ensemble performance. Reweighting is performed dynamically, based on the dataset encoder introduced in [Płudowski et al., 2024] and multilayer perceptron (MLP). This results in dynamic ensembling, opposite to traditional ensembles that operate on static weights. This approach leverages tabular representations captured by the data

Submitted to Table Representation Learning Workshop at NeurIPS 2024. Do not distribute.

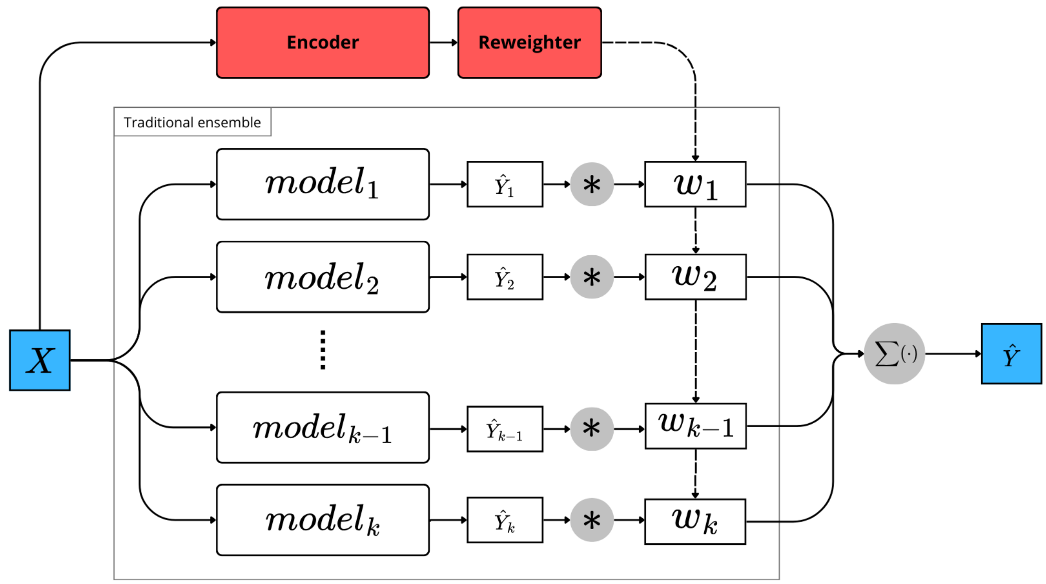


Figure 1: The graphical overview of the framework in binary classification case. The framework aims to boost the performance of the existing ensemble-based model. The innovative elements for the ensemble techniques are highlighted in red. The weights $w_1, \ldots, w_k$ are computed for each observation X using encoder and reweighter elements. The predictions $y_1, \ldots, y_k$ are weighted with $w_1, \ldots, w_k$ and averaged to the final prediction.

encoders to map the observation to the optimal ensemble weights that assure the best prediction. First, all of the models from the ensemble are trained for the new tabular task, just as in the traditional ensemble. Then, the data encoder is fine-tuned to encode each observation from the training set into the space of the ensemble weights to suggest optimal reweighting for specific observations, improving the overall ensemble's performance. The reweighting can be modified to obtain one of several possible desired behaviours of the ensemble: (1) putting most importance on a single model – subspace specialization of the models, (2) sharing importance on the multiple models – robustness by dividing responsibility between models, (3) restricting the change in weights to a small adjustment to static weights – reducing the variance of the encoder element. The graphical overview of the framework is presented in Figure 1.

The proposed framework allows for stably boosting already existing ensembles, potentially improving state-of-the-art solutions for specific task.

The present work offers the following contributions: (1) A new direction based on tabular representation is proposed to improve ensemble methods. (2) A new framework for boosting ensemble – *adaptivee* – is introduced, offering a convenient programming interface for further experiments. (3) Its usefulness is proven on the OpenML-CC18 benchmark in binary classification tasks, with a boost in balanced accuracy of up to 6% compared to the traditional static solutions and boost up to 0.6% compared to state-of-the-art AutoGluon's ensemble methods [Erickson et al., 2020].

2 Related works

Basic methods for creating an ensembling of classifiers use the same set of models for all observations of a dataset. This class of methods is called static ensemble construction methods. Here the Greedy ensemble selection with replacement (GES, Caruana et al. [2004, 2006]) method plays the most important place. Ensembles are built iteratively, starting with an empty set. Then it adds one model that most improves the performance of the collection of models on the ensemble. The big advantage of this solution is the built-in pruning excluding models that do not give satisfactory performance. The simplicity of this approach has made the GES widely applicable in AutoML frameworks Feurer et al. [2019, 2022]. The second static method present in AutoML is stacking, which takes the

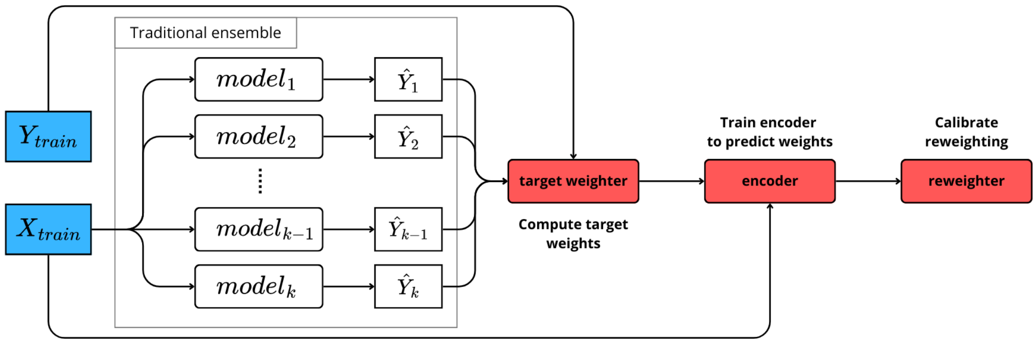


Figure 2: The graphical overview of the *adaptivee* training procedure. The innovative elements for the ensemble techniques are highlighted in red. First, target weighter is used to obtain the target with preferred policy. Then, the encoder is trained based on the X and target weights. Finally, reweighter is optimized to the strategy of choice.

predictions of all underlying models and weights them with a superior algorithm. Its modified version is implemented in AutoGluon [Erickson et al., 2020].

The GES method only takes performance into account causing potential overfitting. Another important aspect affecting the quality of the ensembling is the diversity of the base models, where the diversity of models is understood as making different prediction errors [Hansen and Salamon, 1990]. To address this issue, Boisvert and Sheppard [2021] propose a method for building ensembling using population-based strategies. It turns out that in many cases taking diversity into account ultimately yields an improvement in the quality of ensemblings concerning those built using the GES method [Purucker et al., 2023].

Another approach to ensuring ensembling diversity is dynamic ensembling methods, in which a collection of models is not chosen globally for the entire dataset, but for a single observation [Giacinto and Roli, 2001; Cavalin et al., 2013]. For each observation, a decision must be made as to which models are good enough to use for prediction. It is therefore necessary to explore individual observations and relate the new test observations to those in the training set. Methods differ primarily in this comparison algorithm. The most popular approaches look at a small neighborhood of a given observation and check the accuracy of models in that neighborhood [Woods et al., 1997]. If the model is accurate then it is included in the ensembling. For example, Ko et al. [2008] introduce two strategies KNORA-Eliminate and KNORA-Union which select models that make the correct prediction for all observations or at least one observation located in a given area, respectively. Another approach is to include base models in ensemble is checking their consensus, i.e. ambiguity among the outputs of the classifiers, to predict the true label of the test instance [Dos Santos et al., 2008]. Meta-DES [Cruz et al., 2015] combines and extends these approaches because it considers five different criteria for evaluating the suitability of the underlying model and attaching it to the ensembling.

3 *Adaptivee* framework

The *adaptivee* framework can be split into three main components that extend standard ensemble techniques – creating target weights, encoder and reweighter. Their detailed training overview is presented in Figure 2. The policies that can be used in the framework are summarized in Table 1.

Target weighter. The purpose of the target weighter is to craft the weights on which the encoder should be trained. There are two possible strategies: "Softmax" of inverse errors and "One Takes All". The "Softmax" strategy promotes selecting multiple models to the final prediction and can be described by the following formula:

$$w_i = softmax\left(\frac{1}{\delta|y - \hat{y}|}\right)_i,$$

Table 1: Polices used in the *adaptivee* framework, along with a brief description. During the experiments, all possible combinations of the components are used.

Target Weighter		Encoder		Reweighter	
Softmax	softmax of inversion of prediciton error	**MLP**	simple neural network	**Direction reweight**	weights are moved from the static to dynamic by chosen step
One Takes All	1 for the best model, 0 for others	***liltab***	dataset encoder	**No reweight**	weights from the encoder are passes as is

where w_i is the weight assigned to the i-th model in ensemble and $\hat{y}$ is a vector of predictions made by all models. The hyperparameter δ is used to regularize the weighting; $\delta < 1$ promotes more diverse collection of important models while $\delta > 1$ highlights the most important model. In the contrary to the "Softmax' approach, the "One Takes All" strategy promotes selecting only one model and is described by the following formula:

$$w_i = \mathbb{1}_{\{|y-\hat{y}_i|=\min|y-\hat{y}|\}}.$$

This strategy, in the case of a perfect match for each observation, would select the best model from the considered collection. Consequently, we would obtain maximization of model performance measures based on probability prediction, for example, ROC AUC. On the other hand, this strategy suffers from the highest variance.

Encoder. The encoder part of the pipeline serves the role of the controller that assigns for each observation corresponding weights that should be applied to the ensemble to obtain the best prediction. In the experiments, two models are tested: simple neural network and the *liltab* encoder which is described in Appendix B. While a plain neural network needs to be trained for each task separately, the *liltab* encoder leverages its properties to be pre-trained on the independent data tasks and capture more nuance details in the new tasks.

Reweighter. The reweighter component of the framework is applied to control the variance of the model. As the training of the data encoder that would accurately assign the best ensemble weights for each observation is a challenging task, there is a need to reduce its potential high variance. It is done by the direction reweight that takes static weights w_s, dynamic weights w_d and combine them in final weights w in the following manner:

$$w = w_s + \mu(w_d - w_s),$$

where μ is a hyperparameter to control the impact of the dynamic approach, called the stepsize factor. The static weights w_s can be chosen arbitrarily, e.g., using equal weights for each model or those that are produced by specific AutoML framework like AutoGluon. During our research, we also considered disabling the reweighter component and using only the weights produced by the encoder.

4 Experiments

In the experiments, we consider a portfolio of datasets from OpenML-CC18 [Bischl et al., 2017] constrained to binary tasks. A full list of datasets, altogether with train-test split methodology is presented in Appendix A.

To compare our method to the baseline, we use two static methods to assign weights – equal weights for each model and optimal weights on the training dataset. To find these, we search for maximal accuracy score with the following restrictions (similar to [Iqball and Wani, 2023]):

$$\forall_i w_i \geq 0 \land w_i = pm,\ \sum_{i=1}^{k} w_i = 1,$$

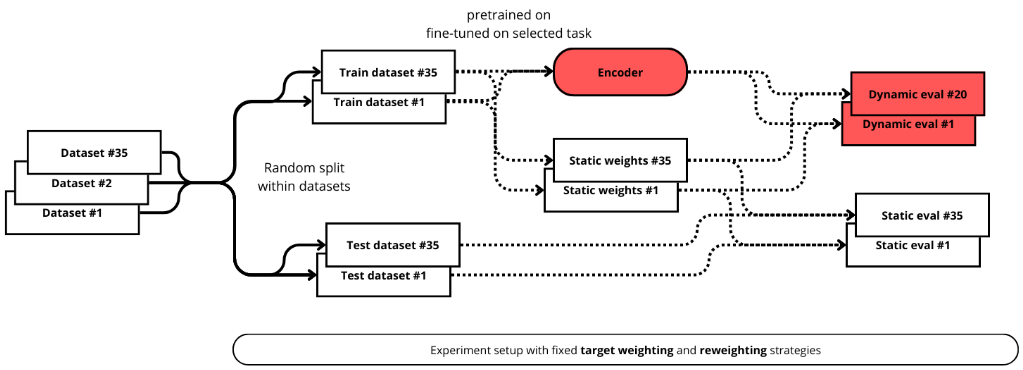


Figure 3: The graphical overview of the experiment setup. Solid lines denote transformation that do not yield new objects, e.g. split data. Dotted lines denote operations that create new objects, e.g., encoder's weights, and evaluation of the experiment. The main contribution of this work is highlighted in red.

where p is a precision of the search and m is arbitrary integer. In our experiments, we set $p = 0.04$ due to the complexity of the search. The models we use in this experiment are listed in Appendix C.1.

In further experiments, we take the best one-layer ensemble model from AutoGluon for each of the datasets and use both the models and their weights as a reference baseline to comparison. The configuration of the AutoGluon we use, altogether with methodology used to retrieve the best ensemble, are listed in Appendix C.2.

The graphical overview of the experiments is presented in Figure 3. First, the best static weights are found for each of the training sets. Then, the encoder is fine-tuned in each of the tasks with one of the possible strategies: "Softmax" or "One Takes All". Finally, the encoder is evaluated on the test set altogether with a reweighting technique. Finally, the results of both static and dynamic ensembles are collected and compared for each of the tasks.

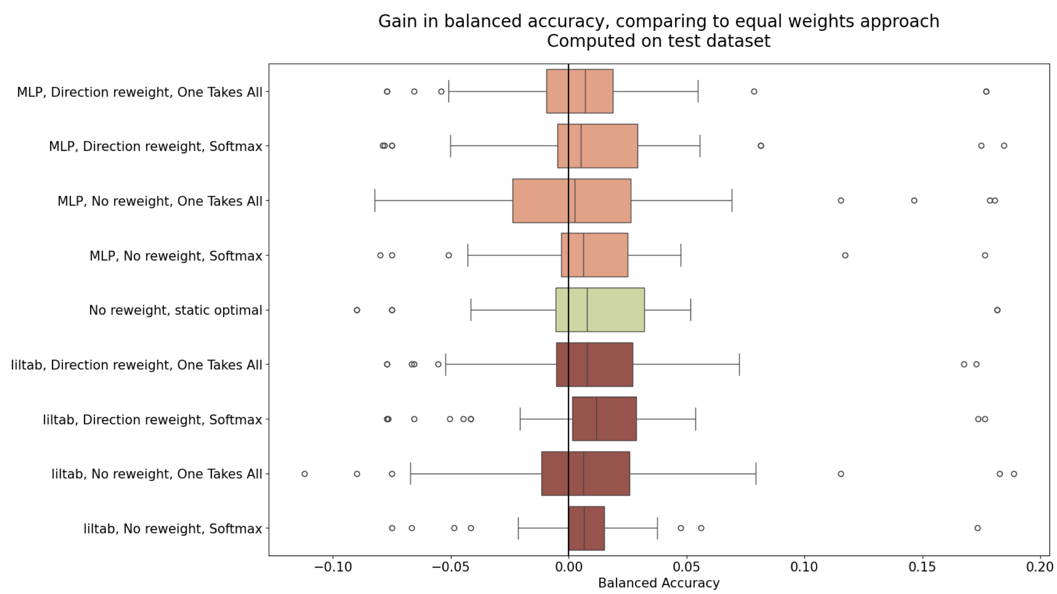


Figure 4: Boxplots of gain over equal weighting static approach. Light orange presents MLP-based dynamic approaches, the red colour denotes *liltab* encoder and light green stands for optimal static approach. In terms of median boost, *adaptivee* framework with "One Takes All" and "Direction Policy" perform best. Approaches based on "Softmax" and no reweighting tend to be better than the equal weighting approach more often than the optimal static approach.

5 Results

In this section, we provide insight into the results of the experiments. First, we present the gain over the the ensemble with equal weighting of each model. This is our baseline as the most common method of creating an ensemble model. Then, we present the overall ranking of all the considered methods. Next, we analyze the framework's ability to produce proper weighting, according to the selected policy. Finally, we use our method to boost the AutoGluon framework to present its usefulness in practical scenarios.

5.1 Fixed portfolio – first scenario

As presented in Figure 4, our framework can outperform the baseline in most tasks. Our best policy combination – including the "Direction Reweight" policy – improves the balanced accuracy by 1.2% at average and by 12% at most. This clearly shows the potential of the dynamic approaches to ensemble weighting. For the "Softmax" policy as target weighter and *liltab* as encoder, over 75% of the tasks performed better compared to the baseline value, creating interesting competition to simple tuning ensemble weights which are presented by the "No reweight, static optimal" method. Additionally, *liltab* encoder achieved a slight increase in the score compared to the MLP encoder, proving its usefulness in the tabular representation task.

The next analysis ranks the *adaptivee* policies along with the static baselines to verify whether the proposed approach is better in most tasks. The results are presented in Figure 5. All considered approaches, excluding equal static weighting, are statistically indistinguishable, yet the proposed framework with the "liltab, Direction reweight, Softmax" policy is better at average compared to the static baselines.

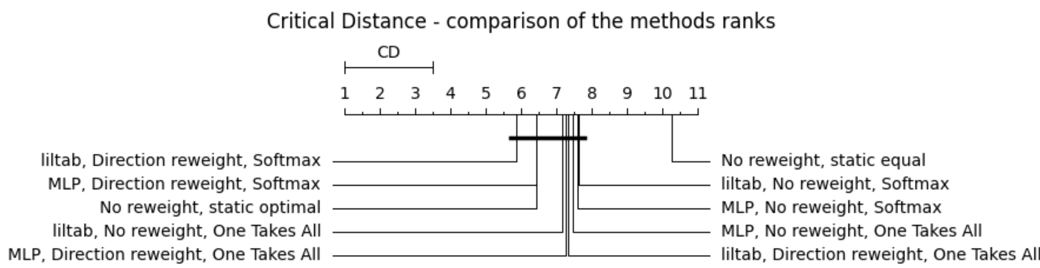


Figure 5: Critical Distance (Critical Difference) plot for Nemenyi test. The test is performed with $\alpha = 0.1$. The critical distance is equal to 2.5 which is close to the difference between the best approach and the equal weighting baseline. The horizontal line shows that there is no statistically significant difference between all considered approaches. However, *adaptivee*-based approaches are more often better compared to static ones.

Next, we verify the difference between static approach and our best dynamic policy. The results are presented in Figure 6. The gain over static approaches is at most equal to 2.8% in balanced accuracy score.

At the end, we perform the ablation study of the quality of proposed weights in *adaptivee* framework. To do so, we examine the L2 distance between the produced by the framework weights and the ones optimal in the selected policy. In Figure 7, it can be observed that the "Softmax" policy is more aligned with its optimal weights compared to "One Takes All". However, the "One Takes All" policy is the optimal one so the overall results of these policies are close to each other, despite the bigger variance of "One Takes All" method.

5.2 AutoGluon portfolio – second scenario

In this experiment, we use the portfolio of models created by the AutoGluon package. As static weights, we take those created by AutoGluon. For simplicity, we present only a comparison to the method that achieved the best increase in the previous experiment – *liltab*-based encoder with "SoftMax" strategy and "Direction reweight". The results are shown in Figure 8. Even though for

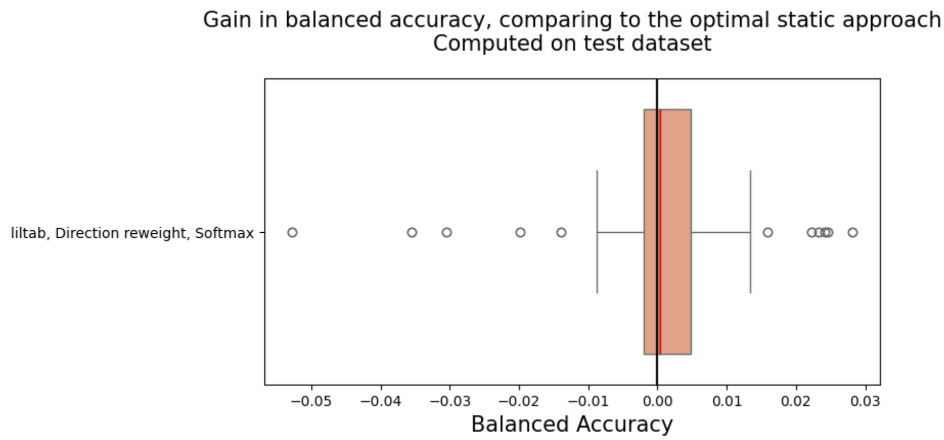


Figure 6: Boxplot of gain over optimal static approach. The red vertical line denotes the average. The average gain is equal to 0.05% in balanced accuracy score.

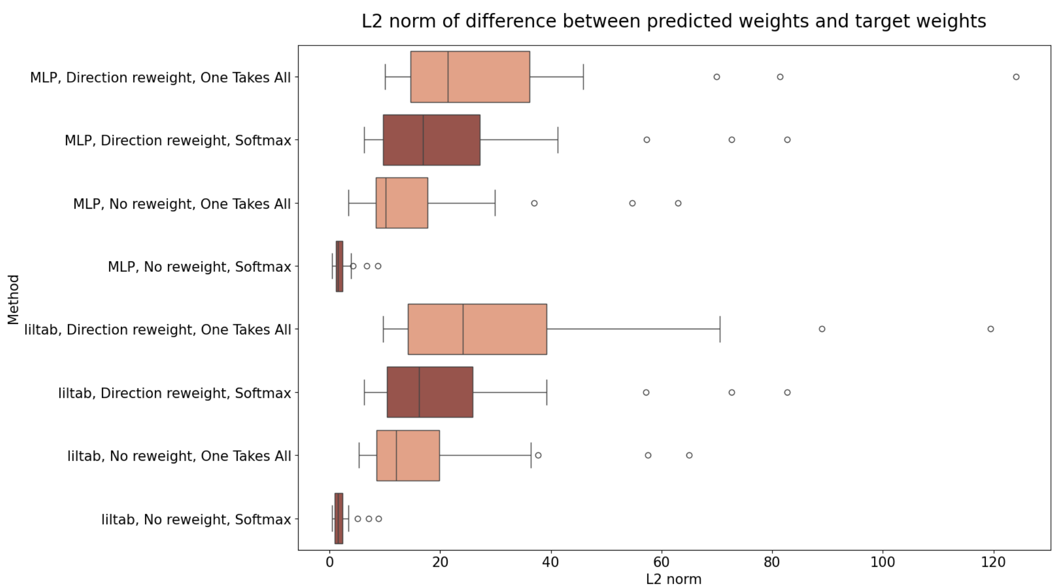


Figure 7: L2 difference between produced and optimal weights in selected policies. While theoretically "One Takes All" should result in a better score than "Softmax", it is harder to learn the encoder its representation.

some datasets applying our approach led to a decrease in performance, for most of them *adaptivee* still increased the value of balanced accuracy, even up to 1%. Moreover, as this evaluation can be easily performed on the validation set, the potential users can easily verify whether *adaptivee* would work in their case.

6 Conclusion

In this article, we propose a new research area for improving the performance of ensemble-based models in tabular data. Following this notion, we implemented and tested the *adaptivee* framework which allows us to boost already existing ensemble-based models up to 6% in the balanced accuracy metric. for state-of-the-art ensemble created by AutoGluon, it leads to a boost of up to 0.6%

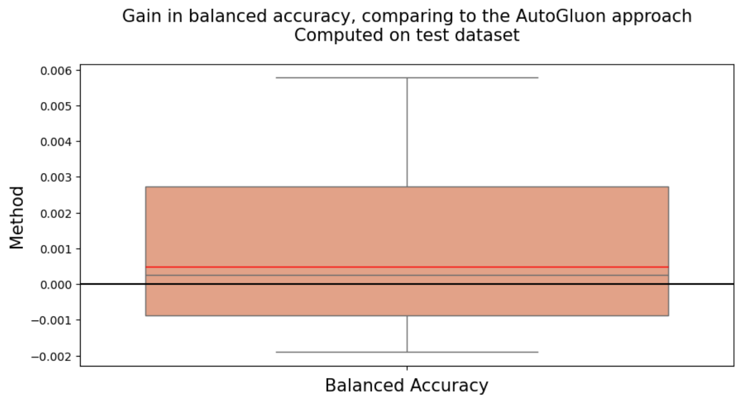


Figure 8: Increase in performance of AutoGluon framework after applying our method on top. The red vertical line denotes the average, while the grey one the median. Our method increases the balanced accuracy score even up to a 0.6% increase compared to state-of-the-art solutions.

in the balanced accuracy. The *adaptivee* framework is currently being developed to serve as an out-of-the-box Python package[1].

Limitation and future work

In this section, we would like to highlight the limitations we recognized during our research. Additionally, we briefly discuss future work in this domain.

The pertaining of the *liltab* encoder in our experiments assumes the same portfolio of the models in all tasks in which the encoder was used. This raises the question of whether the knowledge transfer between tasks holds when the model portfolio is modified. Furthermore, the advantage of the "Direction reweight" policy over "No reweight" emphasizes the fact that the used encoders cannot be applied directly to the task, as they scarcely can effectively suggest the direction of the reweighting, not the exact weights. This should encourage further research on effective ways to create meaningful tabular representation. Finally, proposed policies (i.e. target weighters, encoders and reweighters) do not exhaust all possible combinations and thus, future work should focus on finding new effective strategies to further boost the performance of the ensemble models. In particular, in this work, we did not examine the impact of the hyperparameters of the method. Although important to the practical usage, we did not examine the computation overhead that our solution generates.

References

Bischl, B., Casalicchio, G., Feurer, M., Gijsbers, P., Hutter, F., Lang, M., Mantovani, R. G., van Rijn, J. N., and Vanschoren, J. (2017). Openml benchmarking suites. *arXiv preprint arXiv:1708.03731*.

Boisvert, S. and Sheppard, J. W. (2021). Quality diversity genetic programming for learning decision tree ensembles. In *Genetic Programming: 24th European Conference, EuroGP 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 24*. Springer.

Breiman, L. (2001). Random forests. *Machine learning*.

Cabrera, J. B., Gutiérrez, C., and Mehra, R. K. (2008). Ensemble methods for anomaly detection and distributed intrusion detection in mobile ad-hoc networks. *Information fusion*.

Caruana, R., Munson, A., and Niculescu-Mizil, A. (2006). Getting the most out of ensemble selection. In *Sixth International Conference on Data Mining (ICDM'06)*.

Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. (2004). Ensemble selection from libraries of models. In *Proceedings of the Twenty-First International Conference on Machine Learning*. Association for Computing Machinery.

[1] `https://github.com/DawidPludowski/adaptivee`

Cavalin, P. R., Sabourin, R., and Suen, C. Y. (2013). Dynamic selection approaches for multiple classifier systems. *Neural computing and applications*.

Cruz, R. M., Sabourin, R., Cavalcanti, G. D., and Ren, T. I. (2015). Meta-des: A dynamic ensemble selection framework using meta-learning. *Pattern recognition*.

Dos Santos, E. M., Sabourin, R., and Maupin, P. (2008). A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. *Pattern recognition*.

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

Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., and Hutter, F. (2022). Auto-sklearn 2.0: Hands-free automl via meta-learning. *Journal of Machine Learning Research*.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., and Hutter, F. (2019). *Auto-sklearn: Efficient and Robust Automated Machine Learning*.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*.

Giacinto, G. and Roli, F. (2001). Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition*.

Hansen, L. and Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Iqball, T. and Wani, M. A. (2023). Weighted ensemble model for image classification. *International Journal of Information Technology*.

Iwata, T. and Kumagai, A. (2020). Meta-learning from Tasks with Heterogeneous Attribute Spaces. In *Advances in Neural Information Processing Systems*.

Ko, A. H., Sabourin, R., and Britto Jr, A. S. (2008). From dynamic classifier selection to dynamic ensemble selection. *Pattern recognition*.

Kramer, O. and Kramer, O. (2016). Scikit-learn. *Machine learning for evolution strategies*.

Płudowski, D., Zajko, A., Kozak, A., and Woźnica, K. (2024). Rethinking of encoder-based warm-start methods in hyperparameter optimization. *arXiv preprint arXiv:2403.04720*.

Purucker, L. O., Schneider, L., Anastacio, M., Beel, J., Bischl, B., and Hoos, H. (2023). Q (d) o-es: Population-based quality (diversity) optimisation for post hoc ensemble selection in automl. In *International Conference on Automated Machine Learning*. PMLR.

Rokach, L. (2019). *Ensemble learning: pattern classification using ensemble methods*. World Scientific.

Sesmero, M. P., Ledezma, A. I., and Sanchis, A. (2015). Generating ensembles of heterogeneous classifiers using stacked generalization. *Wiley interdisciplinary reviews: data mining and knowledge discovery*.

Singh, Y., Bhatia, P. K., and Sangwan, O. (2007). A review of studies on machine learning techniques. *International Journal of Computer Science and Security*.

Vanschoren, J., Van Rijn, J. N., Bischl, B., and Torgo, L. (2014). Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*.

Woods, K., Kegelmeyer, W. P., and Bowyer, K. (1997). Combination of multiple classifiers using local accuracy estimates. *IEEE transactions on pattern analysis and machine intelligence*.

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC press.

A Data

In our experiments, we used OpenML-CC18 [Bischl et al., 2017]. As a result, we got 35 tasks. The "Train" set was used to pre-train the *liltab* encoder (due to the non-heterogeneous nature of the MLP model, it was not pre-trained). In this case, the target weights were calculated with the "Softmax" policy on the whole dataset. The datasets were at random split into train and test parts with a ratio of 3:1. All randomness in described operations is controlled by the seed to ensure reproducible results. In Table 2, we present the summary of the used data.

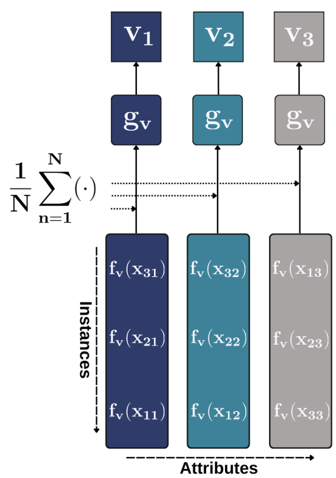
Table 2: Datasets used during the experiments. The source of the data is OpenML repository [Vanschoren et al., 2014].

Dataset	Set
adult	Train-test
Banknote-authentication	Train-test
Bioresponse	Train-test
Breast-w	Train-test
Cylinder-bands	Train-test
diabetes	Train-test
electricity	Train-test
ilpd	Train-test
madelon	Train-test
ozone-level-8hr	Train-test
pc3	Train-test
PhishingWebsites	Train-test
sick	Train-test
Bank-marketing	Train-test
Blood-transfusion-service	Train-test
churn	Train-test
Climate-model-simulation	Train-test
Credit-approval	Train-test
Credit-g	Train-test
Dresses-sales	Train-test
Internet-advertisements	Train-test
jm1	Train-test
kc1	Train-test
kc2	Train-test
Kr-vs-kp	Train-test
pc1	Train-test
pc4	Train-test
phoneme	Train-test
nomao	Train-test
Qsar-biodeq	Train-test
smapbase	Train-test
numerai28.6	Train-test
Tic-tac-toe	Train-test
wdc	Train-test
wilt	Train-test

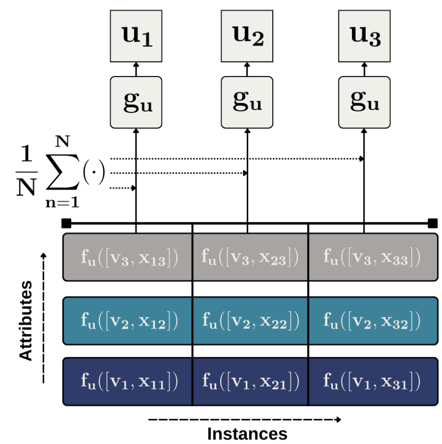
B Heterogeneous data encoder – *liltab*

Here, we provide a brief introduction to *liltab* architecture. For further reading, please see [Płudowski et al., 2024] for dataset encoder details and [Iwata and Kumagai, 2020] for the encoder inspiration.

Network Architecture. The *liltab* architecture is inspired by [Iwata and Kumagai, 2020]. It encodes datasets through a neural network structure. The architecture processes the input data to extract representations that capture marginal distributions and relationships between attributes and target variables. It consists of two main components which are presented in Figure 9. In the diagrams, $f_{\odot}$ and $g_{\odot}$ are feed-forward neural networks; $f_{\odot}$ are specialized to encode information from the single elements from the input, while $g_{\odot}$ are focused on summarizing the information. In the first step, the initial representation v is obtained. Next, it is used the create the final representation u which is easily transformed into weights $w = (w_1, \ldots, w_k)$ observation-wise. Because of the first part which encodes the marginal distribution, the *liltab* encoder is supposed to perform better on predicting weights of bigger batches as it allows for better capture of the data distribution.



(a) First step of the *liltab* encoder. Here, the inference network learns about the empirical marginal distributions of the attributes based on the provided data subset.



(b) Second step of the *liltab* encoder. Here, the inference network learns the relationships between the attributes based on the provided data subset. To ensure weights are summed up to 1, neural network g_u applies softmax function to create final representation.

Figure 9: Overview of the *liltab* architecture. The diagrams come from [Płudowski et al., 2024], with small adjustments.

Training Process. In the original work, the *liltab* network is trained using a contrastive learning approach. In our work, however, we modified its learning process to capture tabular representation that can be used to retrieve the weights for an ensemble model. To do this, we treated the weights from the Target Weighter component as a ground truth Y. Similarly to the traditional supervised learning, the set of observations is treated as X.

C Models

Here, we provide details about the model portfolio that we used during the experiment part of this article. First, we list the models used while testing a fixed portfolio of models and then, we describe the configuration of the AutoGluon framework which we use throughout the second part of the experiments.

C.1 Fixed models portfolio – first scenario

In this experiment, we used the portfolio of models listed in Table 3. All of the models were trained using the default hyperparameter values and no tuning was performed. Although this approach may be considered a bad practice, we argue that in our method we aim to boost the performance of any ensemble of the models, regardless of its initial score in any metric. Moreover, the analysis of the best possible models is performed in the second part of the experiments.

Table 3: Models used during the experiments. All implementation was taken from the Sci-kit Learn package [Kramer and Kramer, 2016].

Model	
Model Name	Model Class
logistic regression	LogisticRegression
decision tree	DecisionTreeClassifier
LDA	LinearDiscriminantAnalysis
naive Bayes	GaussianNB
random forest	RandomForestClassifier
K-nearest neighbours	KneighborsClassifier

C.2 AutoGluon models portfolio – second scenario

In this experiment, we used `TabularPredictor` class from AutoGluon package to create the portfolio of models, altogether with the corresponding weights. To simplify the evaluation, we restricted ourselves to using only a one-layer ensemble. Moreover, we force using Bootstrap to produce a more versatile portfolio of models. The full parametrization of the `predictor.fit` is listed in Table 4. Please note that most training took less than the time limit specified by `time_limit` so the models used in this experiment may be treated as the best possible with restriction to the parametrization.

For three datasets, the AutoGluon framework failed to create an ensemble (in fact, it produced an ensemble object containing only one model). We decided to omit these datasets in the final results as they create ties in score that are not justified by the method performance.

Table 4: The parametrization of the AutoGluon framework [Erickson et al., 2020].

Autogluon	
Parameter	Value
`num_stack_levels`	0
`num_bag_sets`	2
`num_bag_folds`	5
`time_limit`	300

D Experiments hardware details

In Table 5 we present the total time required to run all experiments that we showed in the "Results" section. The total time required to explore specific strategies and manually select hyperparameters is not reported but is estimated to be circa 50 hours on the provided hardware.

Table 5: Hardware and time specification of the experiments provided in the article.

Experimental Setup			
Operation	File name	Hardware	Time
Downloading and preprocessing data	bin/download_openml_data.py	Intel core vPRO i9, RTX 3080, 48GB RAM	0.5h
Pretraining of the encoder	bin/prepare_liltab_data.py		1h
Experiments – first scenario	bin/run_analysis.py		10h
Experiments – second scenario	bin/run_autogluon_analysis.py		3h

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main contribution is stated at the end of the introduction. All three claims in the contribution is explained and justified in the "Methodology" and "Results" sections.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: All recognized limitations are presented at the end of the article. The limitations are formulated to create a list of future work that needs to be performed to make the research truly mature enough to make stronger claims than those provided in the "Introduction" section.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include any theoretical work. All formulas presented in the paper serve only as support to present the code of the framework in a mathematical manner.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: the reproducibility of the experiments can be easily done by following the information contained in `README.md` file in the GitHub repository mentioned in the main part of the article. Moreover, a high-level explanation of the experiments is provided in the article text (excluding details like random seeds used during the training).

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The referenced repository (in the footer) contains instruction (`README.md`) with all necessary code that needs to be executed to reproduce the results. Obtaining final results requires running a few `bash` code lines and notebooks.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The main parametrization is provided in the appendices. The detail parametrization is a part of the code (default values of the classes/functions and parameters provided in the `bin` directory).

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: The plots provided in the "Results" section presents statistical significance of the main results (Critical Distance plots). The p-value is provided. Other results are presented in the form of box plots which present all statistically important aspects of the results. All presented values are obtained from the test sets of the data.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: the summary of the time required for all of the experiments is specified in Appendix D.

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: After careful verification, we do not see any aspects of our work that could harm the NeurIPS Code of Ethics.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We do not discuss the societal impacts of our work as we believe that this paper does not introduce any methods or ideas that could have any significant impact on society.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This work does not introduce any dataset that could pose any harm. The proposed tabular representation encoder cannot be directly used for any application that could pose a negative or non-ethical impact.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite every paper that introduces the asses that we used during our research, which refers mostly to the Python packages and datasets. If anything was omit by accident in citations, it is still in requirements file on the repository.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: this article does not introduce any new asset (release of the pretrained encoder proposed in this paper is planned to be done after successful submission).

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: the paper does not involve crowdsourcing nor research with human subjects.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.