In-Context Meta-Learning with Large Language Models for Automated Model and Hyperparameter Selection

Anonymous Author(s)
Affiliation
Address
email

Abstract

Model and hyperparameter selection is a critical yet costly step in machine learning, often requiring expert intuition or extensive search. We investigate whether large language models (LLMs) can reduce this cost by acting as in-context meta-learners that generalize across tasks to propose effective model-hyperparameter choices without iterative optimization. Each task is represented as structured metadata, and we prompt an LLM under two strategies: Zero-Shot, using only the target task metadata, and Meta-Informed, which augments the prompt with metadata–recommendation pairs from prior tasks. Evaluated on 22 tabular Kaggle challenges, Meta-Informed prompting outperforms Zero-Shot and hyperparameter optimization baselines, approaching expert AutoML blends while yielding interpretable reasoning traces and efficiency gains under tight training budgets. These results suggest that LLMs can transfer knowledge across tasks to guide automated model selection, establishing model and hyperparameter selection as a concrete testbed for studying emergent adaptation beyond language domains.

1 Introduction

As large language models (LLMs) scale, they increasingly exhibit emergent behaviors allowing them to adapt to new tasks by reusing patterns from prior experience provided in-context [Brown et al., 2020, Dong et al., 2024]. Studying such behaviors outside of language tasks is key to understanding their scope and reliability. In this work, we use model and hyperparameter selection as a testbed for evaluating whether LLMs can perform in-context meta-learning.

Performance in machine learning depends heavily on choosing model families and hyperparameters, known as the Combined Algorithm Selection and Hyperparameter optimization (CASH) problem [Thornton et al., 2013]. Conventional methods such as grid search and Bayesian optimization are costly and knowledge-intensive. If LLMs can generalize across tasks in this context, it would not only aid AutoML but also provide a concrete setting for evaluating cross-task adaptation.

Our approach consists in representing each task with structured metadata (e.g. sample size, dimensionality, feature types) and prompts an LLM to output a candidate configuration model class (e.g. LGBM, MLP) and hyperparameters. We consider two prompting strategies: **Zero-Shot**, using only the target metadata, and **Meta-Informed**, which augments the prompt with metadata-configuration pairs from prior tasks (Figure 1).

Contributions. (1) We show that LLMs can address the CASH problem in-context by mapping task metadata to model and hyperparameter configurations. (2) On 22 Kaggle datasets under limited budgets, we find that Meta-Informed prompting outperforms Zero-Shot and Hyperopt baselines, with reasoning traces revealing how LLMs connect dataset characteristics to prior tasks.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

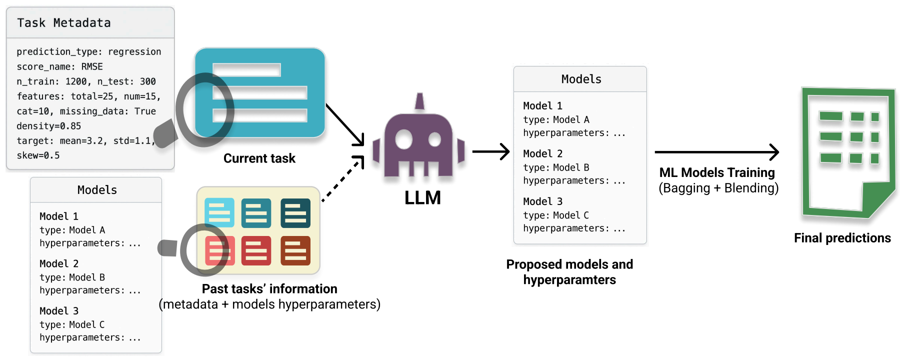

Figure 1: Overview of the method. Each task is represented by metadata, and the LLM outputs model and hyperparameter configurations. The dotted arrow indicates the inclusion of prior-task metadata-configuration pairs in the **Meta-Informed** setting.

Together, these results provide a compact case study of emergent LLM capabilities in a domain beyond language.

A more detailed discussion of related work on hyperparameter optimization, meta-learning, CASH, and recent LLM-based methods is provided in Appendix A.

2 Methodology and Results

2.1 Method

We formulate model and hyperparameter selection as an in-context meta-learning task. Each dataset is summarized by a structured metadata block describing high-level properties such as prediction type, evaluation metric, sample sizes, feature composition, missingness, and target statistics (see Appendix D.1).

On each Kaggle challenge, the LLM is prompted to propose model-hyperparameter ensembles under two modes:

- **Zero-Shot:** only the metadata of the target task is provided.
- **Meta-Informed:** the prompt additionally includes reference metadata-configuration pairs from prior tasks.

For the reference pool used in the **Meta-Informed** strategy context, we extract **Context Blends** from the top 10 contributors (by ensemble weight) of AutoML-generated blends obtained via extensive hyperparameter search.

Details of the base models are provided in Appendix F. Our experiments use the DeepSeek-R1 model [DeepSeek-AI et al., 2025], with prompt design and LLM configuration described in Appendices D and, respectively E. Each seed corresponds to a new run with a different set of **Context Blends**.

2.2 Datasets

We evaluate our method on 22 Kaggle tabular challenges spanning both regression and classification. The benchmark includes a mix of "playground" competitions (synthetic or repurposed datasets) and "featured" challenges (real industrial or scientific applications). Prediction types range from regression to binary and multi-class classification, with metrics including error-based losses (RMSE, MAE, RMSLE), probabilistic measures (AUC, log-loss, NLL), and discrete scores (accuracy, F_1). Dataset scales vary widely from fewer than 2,000 training points (`horses`) to several hundred thousand (`media`, `insurance`) while feature dimensionality ranges from fewer than 10 (`abalone`) to over a thousand (`molecules`). This diversity ensures coverage of small vs. large data regimes, low- vs. high-dimensional settings, and synthetic vs. real-world tasks. Full dataset details are provided in Table 2 in the Appendix.

2.3 Performance Comparison

To assess the quality of LLM-generated ensembles, we compare them against several standard baselines that capture random selection and conventional hyperparameter optimization. For fairness, all methods are allowed to train exactly 10 models. Specifically, we evaluate three baselines (detailed in Appendix H): **Context-Random** (uniformly samples k=10 model-hyperparameter configurations from the same context to test whether LLMs provide value beyond random reuse), **Random-Hyperopt** (runs 10 iterations of hyperopt with a uniformly sampled model family), and **LGBM-Hyperopt** (also runs 10 iterations of hyperopt but restricted to LightGBM, reflecting the strength of a single well-tuned model). Both Hyperopt-based baselines are implemented with HEBO [Cowen-Rivers et al., 2022], one of the most effective and consistent hyperparameter optimization methods across a wide range of tasks [Kegl, 2023].

Results. Blend quality is measured using the private leaderboard percentile rank (p-rank; higher is better) after training on the Kaggle datasets. Figure 2 summarizes the average performance across 22 datasets. **Meta-Informed** achieves the strongest LLM-driven performance (72.7), surpassing both **Zero-Shot** (70.4) and **Context-Random** (70.0), while clearly outperforming **Random-Hyperopt** (65.7). Although the AutoML-derived **Context Blends** remains higher (77.7), the gap is modest given that no iterative search is performed, showing that LLMs can interpret metadata and make competitive recommendations. Importantly, the significant improvement of **Meta-Informed** over **Context-Random** indicates that the LLM is not merely sampling from the metadata, but is leveraging past tasks' information in a way that reflects genuine adaptation.

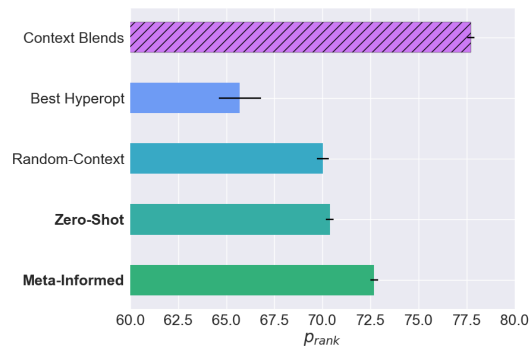

Figure 2: Comparison of prompting strategies and baselines in terms of p_{rank}. The **Context Blends** produced by AutoML performance for each challenge are shown as a reference. Error bars indicate 90% confidence intervals of the mean across 8 random seeds per dataset.

Looking at the detailed per-challenge results (Table 1) alongside the dataset metadata (Table 2), we observe that performance patterns vary across tasks. The strongest improvements of the Meta-Informed LLM appear on datasets with many samples and mixed feature types, such as `mental health` (140k samples, categorical and numerical features), `insurance` (300k samples, mixed features), and `housing california` (37k samples, purely numerical). By contrast, on relatively low-dimensional regression benchmarks such as `abalone` and `concrete strength`, the benefit is less consistent, and baselines can perform better. We also note that while **LGBM-Hyperopt** has the lowest mean score overall, it performs strongly on certain tasks (e.g., `loan approval`), likely benefiting from restricting search to a single competitive model family. Finally, across most datasets, LLM-based methods exhibit lower variance than Hyperopt baselines, indicating more stable performance.

2.4 Performance Efficiency

To complement performance ranking, we also evaluate efficiency relative to standard hyperparameter optimization. For this comparison, we focus on a subset of six datasets: `abalone`, `blueberry`, `covertype`, `heat flux fi`, `horses`, and `media`.

We define one *round* as training a single model configuration followed by its integration into the blending pipeline, ensuring all methods incur the same per-round cost. The LLM based methods produce exactly ten configurations in a single forward pass, after which no additional training is performed. By contrast, Hyperopt continues to propose new configurations sequentially.

We consider two model selection variants (see Appendix H for details): **Random-Hyperopt**, which runs 10 iterations of HEBO on a uniformly sampled model family, and **MaxUCB-Hyperopt**, which follows Balef et al. [2025] by treating each family as a bandit arm and selecting the arm that maximizes an upper-confidence bound before applying HEBO within that family.

Kaggle Challenge	Meta-Informed	Zero-Shot	Context-Random	Random-Hyperopt	LGBM-Hyperopt
abalone	85.73 ± 3.3	74.67 ± 4.6	**87.87 ± 2.3**	58.95 ± 4.6	64.21 ± 11.3
allstate	**69.92 ± 2.3**	61.66 ± 2.9	65.41 ± 5.0	50.05 ± 2.4	51.0 ± 2.7
attrition	59.51 ± 1.7	**61.12 ± 1.8**	57.31 ± 2.3	59.36 ± 3.3	48.21 ± 5.0
blueberry	**81.16 ± 2.4**	79.86 ± 1.7	78.96 ± 3.8	70.77 ± 5.3	65.87 ± 7.7
churn	70.35 ± 0.9	68.73 ± 0.9	68.71 ± 3.0	65.07 ± 4.0	**70.64 ± 1.0**
cirrhosis	70.58 ± 3.6	69.09 ± 1.4	**73.06 ± 1.8**	64.61 ± 4.6	70.17 ± 2.0
concrete strength	74.34 ± 17.9	74.19 ± 6.8	59.37 ± 16.1	88.81 ± 5.4	**83.21 ± 9.3**
covertype	**67.78 ± 4.0**	58.35 ± 7.6	60.05 ± 10.3	56.75 ± 11.0	32.0 ± 3.4
crab age	**68.87 ± 0.7**	68.81 ± 0.6	67.67 ± 1.2	61.84 ± 2.3	63.84 ± 1.8
credit fusion	96.61 ± 1.0	96.71 ± 1.1	90.91 ± 1.7	96.35 ± 0.9	**96.75 ± 1.5**
failure	41.12 ± 1.5	43.52 ± 1.7	41.25 ± 0.8	43.7 ± 2.6	**48.15 ± 7.0**
heat flux fi	**93.4 ± 5.0**	90.7 ± 4.3	83.65 ± 8.6	69.07 ± 6.6	36.22 ± 17.1
horses	82.39 ± 7.7	**82.78 ± 5.6**	75.31 ± 10.6	81.15 ± 6.2	79.75 ± 5.7
housing california	**62.53 ± 0.6**	54.84 ± 2.4	60.07 ± 2.0	46.9 ± 6.8	52.71 ± 3.9
influencers	76.84 ± 7.4	83.55 ± 1.4	80.52 ± 2.8	82.95 ± 2.7	**87.45 ± 1.9**
insurance	**74.68 ± 2.4**	68.16 ± 1.8	67.9 ± 2.1	62.53 ± 5.9	64.6 ± 3.4
loan approval	71.58 ± 2.6	63.29 ± 5.5	66.84 ± 5.4	62.64 ± 6.9	**74.43 ± 0.9**
media	**62.95 ± 1.4**	57.52 ± 2.0	61.81 ± 2.5	49.5 ± 7.5	26.07 ± 2.8
mental health	**92.99 ± 3.0**	79.77 ± 10.2	89.69 ± 5.2	75.34 ± 9.5	80.11 ± 7.7
mercedes	17.81 ± 2.8	36.44 ± 7.8	35.26 ± 10.6	**36.57 ± 8.6**	25.42 ± 2.0
molecules	**97.52 ± 1.5**	96.34 ± 1.6	96.32 ± 3.3	96.33 ± 2.6	78.02 ± 12.6
unknown a	**80.56 ± 0.8**	78.6 ± 0.8	72.59 ± 2.4	66.17 ± 2.5	61.41 ± 5.5
Mean	**72.69 ± 0.2**	70.39 ± 0.2	70.02 ± 0.3	65.7± 1.1	61.8± 1.1

Table 1: Kaggle private leaderboard percentile rank (p-rank) across 22 challenges (higher is better). Uncertainty is reported as ± values, representing the 90% confidence interval based on standard error across 8 random seeds. Full results including context blends performance are given in Appendix B.2.

On these six datasets, the LLM based methods match or exceed Hyperopt performance within the same budget of ten training rounds, while Hyperopt seems to require substantially more rounds to achieve similar performance (Figure 3). In practice, LLMs may be even more advantageous since they generate all configurations in a single pass rather than sequentially.

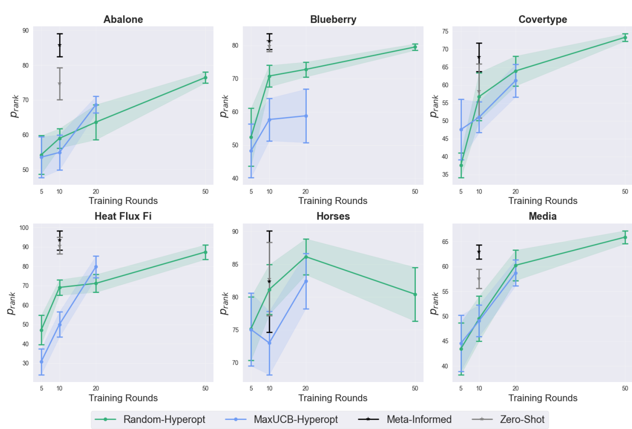

Figure 3: p_{rank} over training rounds for **Random-Hyperopt**, **MaxUCB-Hyperopt**, **Meta-Informed**, and **Zero-Shot** across the six selected datasets. Error bars indicate 90% confidence intervals using standard error across 8 seeds.

2.5 Interpretability

Another advantage of LLM-based methods is interpretability. Unlike conventional hyperparameter optimization, which produces configurations without explanation, the LLM generates structured outputs accompanied by reasoning traces. These traces highlight how the model can relate task metadata to past examples when proposing new model-hyperparameter ensembles. For example, the LLM often explains its choices by linking dataset properties to its choices such as favoring CatBoost on feature sets dominated by categorical variables, or suggesting deeper trees when the regression task involves many numeric features. Appendix G presents selected reasoning traces that illustrate how the model draws on prior tasks and/or its internal knowledge to guide model and hyperparameter recommendations.

3 Conclusion

We evaluated LLM-based prompting for CASH on 22 Kaggle challenges. **Meta-Informed** prompting consistently outperforms **Zero-Shot** and Hyperopt baselines, though it remains below expert AutoML blends. LLM-generated ensembles offer efficiency under limited budgets and interpretability via reasoning traces. These results suggest that LLMs can accelerate and guide AutoML while demonstrating cross-task adaptability as a form of in-context meta-learning.

References

Sungyong Baik, Myungsub Choi, Janghoon Choi, Heewon Kim, and Kyoung Mu Lee. Meta-learning with adaptive hyperparameters. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Amir Rezaei Balef, Claire Vernade, and Katharina Eggensperger. Put cash on bandits: A max k-armed problem for automated machine learning, 2025. URL `https://arxiv.org/abs/2505.05226`.

James Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305, 03 2012.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL `https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf`.

Giorgos Borboudakis, Paulos Charonyktakis, Konstantinos Paraschakis, and Ioannis Tsamardinos. A meta-level learning algorithm for sequential hyper-parameter space reduction in automl. *arXiv preprint arXiv:2312.06305*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 18, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138385. doi: 10.1145/1015330.1015432. URL `https://doi.org/10.1145/1015330.1015432`.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL `http://doi.acm.org/10.1145/2939672.2939785`.

Alexander Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan-Rhys Griffiths, Alexandre Maravel, Jianye Hao, Jun Wang, Jan Peters, and Haitham Bou Ammar. Hebo: Pushing the limits of sample-efficient hyperparameter optimisation. *Journal of Artificial Intelligence Research*, 74, 07 2022.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A survey on in-context learning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.64. URL `https://aclanthology.org/2024.emnlp-main.64/`.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper_files/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf`.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 1487–1495, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. doi: 10.1145/3097983.3098043. URL `https://doi.org/10.1145/3097983.3098043`.

Yi-Qi Hu, Xu-Hui Liu, Shu-Qiao Li, and Yang Yu. Cascaded algorithm selection with extreme-region ucb bandit. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 07 2021. doi: 10.1109/TPAMI.2021.3094844.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 240–248, Cadiz, Spain, 09–11 May 2016. PMLR. URL `https://proceedings.mlr.press/v51/jamieson16.html`.

Balazs Kegl. A systematic study comparing hyperparameter optimization engines on tabular data, 2023. URL `https://arxiv.org/abs/2311.15854`.

Roman Kochnev, Arash Torabi Goodarzi, Zofia Antonina Bentyn, Dmitry Ignatov, and Radu Timofte. Optuna vs code llama: Are llms a new paradigm for hyperparameter tuning?, 2025. URL `https://arxiv.org/abs/2504.06006`.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: a novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18(1): 6765–6816, January 2017. ISSN 1532-4435.

Yang Li, Jiang Jiawei, Jinyang Gao, Yingxia Shao, Ce Zhang, and Bin Cui. Efficient automatic cash via rising bandits. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:4763–4771, 04 2020. doi: 10.1609/aaai.v34i04.5910.

Siyi Liu, Chen Gao, and Yong Li. AgentHPO: Large language model agent for hyper-parameter optimization. In *The Second Conference on Parsimony and Learning (Proceedings Track)*, 2025. URL https://openreview.net/forum?id=HU3yfXcoKU.

Kanan Mahammadli and Seyda Ertekin. Sequential large language model-based hyper-parameter optimization, 2025. URL https://arxiv.org/abs/2410.20302.

Neeratyoy Mallik, Eddie Bergman, Carl Hvarfner, Danny Stoll, Maciej Janowski, Marius Lindauer, Luigi Nardi, and Frank Hutter. Priorband: Practical hyperparameter optimization in the age of deep learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=uoiwugtpCH.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Valerio Perrone, Rodolphe Jenatton, Matthias W. Seeger, and C. Archambeau. Scalable hyper-parameter transfer learning. In *Neural Information Processing Systems*, 2018. URL https://api.semanticscholar.org/CorpusID:54035096.

Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *NeurIPS*, pages 6639–6649, 2018. URL http://dblp.uni-trier.de/db/conf/nips/nips2018.html#ProkhorenkovaGV18.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/f33ba15effa5c10e873bf3842afb46a6-Paper.pdf.

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 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*, KDD '13, page 847–855, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450321747. doi: 10.1145/2487575.2487629. URL https://doi.org/10.1145/2487575.2487629.

Ying Wei, Peilin Zhao, and Junzhou Huang. Meta-learning hyperparameter performance prediction with neural processes. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11058–11067. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/wei21c.html.

Michael R. Zhang, Nishkrit Desai, Juhan ae, Jonathan Lorraine, and Jimmy. Using large language models for hyperparameter optimization, 2024. URL https://arxiv.org/abs/2312.04528.

Mingkai Zheng, Xiu Su, Shan You, Fei Wang, Chen Qian, Chang Xu, and Samuel Albanie. Can gpt-4 perform neural architecture search?, 04 2023.

Appendix

A Related work

Hyperparameter Optimization Early hyperparameter optimization (HPO) techniques included simple search strategies such as grid search and random search [Bergstra and Bengio, 2012]. More sophisticated model-based methods include Bayesian optimization (e.g., Gaussian process-based BO) which iteratively fits a surrogate model to past evaluations [Bergstra et al., 2011, Snoek et al., 2012]. Multi-fidelity and bandit-based approaches, such as Hyperband [Li et al., 2017] and Successive Halving [Jamieson and Talwalkar, 2016], exploit early-stopping to allocate resources efficiently. Subsequent extensions incorporate problem structure: for instance, compute-aware or multi-task Bayesian optimization methods transfer information across related tasks [Swersky et al., 2013, Golovin et al., 2017].

Meta-Learning and HPO Meta-learning-based hyperparameter optimization methods aim to generalize optimization strategies across tasks by leveraging prior experience. Transfer Neural Processes (TNP) [Wei et al., 2021], for example, incorporate meta-knowledge such as surrogate models and historical trial data to enhance sample efficiency. Meta-Bayesian optimization methods extend this idea by using priors over surrogate models learned from related tasks, enabling faster convergence in new optimization problems [Feurer et al., 2015, Perrone et al., 2018]. Other approaches, such as ALFA [Baik et al., 2020], learn to adapt hyperparameters dynamically during training, modeling the optimization process itself. Techniques like SHSR [Borboudakis et al., 2023] improve efficiency by pruning unpromising regions of the search space using past AutoML runs. PriorBand [Mallik et al., 2023] further accelerates HPO by combining expert beliefs with low-fidelity proxy tasks to guide the search in deep learning pipelines.

The CASH Problem The problem of jointly searching the model class and its hyperparameters has been coined the Combined Algorithm Selection and Hyperparameter optimization (CASH) problem [Thornton et al., 2013]. A common approach is to treat model choice as a categorical hyperparameter and perform HPO directly over the hierarchical space of algorithms and their parameters. AutoML systems such as Auto-WEKA and Auto-sklearn adopt this combined-search strategy [Thornton et al., 2013, Feurer et al., 2015], but the hierarchical and high-dimensional nature of these spaces makes optimization challenging. Running separate HPO procedures per model class is computationally prohibitive and scales poorly with the number of candidate algorithms. To mitigate these limitations, recent work has proposed decomposed CASH formulations, where algorithm selection is cast as a bandit problem and hyperparameter optimization is performed within each arm. In particular, Balef et al. [2025] introduce MaxUCB, a Max-armed bandit algorithm designed for the light-tailed, bounded, and left-skewed reward distributions characteristic of HPO, showing improved efficiency compared to classical combined search and competing bandit variants such as Rising Bandits [Li et al., 2020] and ER-UCB [Hu et al., 2021]. Unlike both combined and decomposed approaches, our method leverages an LLM to tackle the CASH problem directly in a zero-shot manner, jointly suggesting both model families and hyperparameters without requiring bandit-style exploration or expensive hierarchical search.

LLM-Based HPO Recent work has explored the use of LLMs for hyperparameter optimization in ML tasks. Zhang et al. [2024] showed that LLMs can generate effective hyperparameters by iteratively refining suggestions and incorporating feedback, achieving results comparable to traditional methods such as Bayesian optimization. Kochnev et al. [2025] showed that a fine-tuned Code Llama model can suggest hyperparameters for neural networks from code descriptions, outperforming tools like Optuna in a few trials, while Zheng et al. [2023] demonstrated that LLMs can be used to find competitive architectures on neural architecture search benchmarks. Mahammadli and Ertekin [2025] introduced a hybrid approach combining LLMs with Bayesian optimization, showing improved performance on tabular classification tasks. Liu et al. [2025] proposed AgentHPO, where an LLM autonomously designs and refines experiments based on task descriptions, performing competitively with expert-tuned configurations. However, these methods primarily focus on hyperparameter tuning in isolation, leaving the broader CASH problem unaddressed. In contrast, our method operates in a purely zero-shot setting and addresses CASH directly, achieving competitive results without requiring iterative feedback or access to validation performance during inference, while still leveraging prior task information for cross-task generalization in the meta-informed setting.

B Kaggle Benchmark Details

B.1 Kaggle Challenges

Table 2 summarizes the statistics of the tabular challenges used in this paper, highlighting a wide range of problem types, metrics, and data sizes.

Kaggle challenge	**type**	**year**	**pred type**	**metric**	**# team**	**# train**	**# test**	**# feat**	**# cat**	**# num**	**# cls**	**# miss**
abalone	play	2024	reg	rmsle	2606	90615	60411	8	1	7		0
allstate	feat	2016	reg	mae	3045	188318	125546	130	116	14		0
attrition	play	2023	bin	auc	665	1677	1119	33	8	25	2	0
blueberry	play	2023	reg	mae	1875	15289	10194	16	0	16		0
churn	play	2024	bin	auc	3632	165034	110023	12	6	6	2	0
cirrhosis	play	2023	mult	nll	1661	7905	5271	18	6	12	3	0
concrete strength	play	2023	reg	rmse	765	5407	3605	8	0	8		0
covertype	play	2015	mult	acc	1692	15120	565892	54	44	10	7	0
crab age	play	2023	reg	mae	1429	74051	49368	8	1	7		0
credit fusion	feat	2011	bin	auc	924	150000	101503	10	0	10	2	56384
failure	play	2022	bin	auc	1888	26570	20775	24	3	21	2	35982
heat flux fi	play	2023	reg	rmse	693	21229	10415	8	2	6		34603
horses	play	2023	bin	f1	1541	1235	824	27	17	10	3	1324
housing california	play	2023	reg	rmse	689	37137	24759	8	0	8		0
influencers	feat	2013	bin	auc	132	5500	5952	22	0	22	2	0
insurance	play	2021	reg	rmse	1433	300000	200000	24	10	14		0
loan approval	play	2024	bin	auc	3858	58645	39098	11	4	7	2	0
media	play	2023	reg	rmsle	952	360336	240224	15	7	8		0
mental health	play	2024	bin	acc	2685	140700	93800	18	7	8	2	718167
mercedes	feat	2017	reg	r2	3823	4209	4209	376	376	0		0
molecules	feat	2012	bin	nll	698	3751	2501	1776	0	1776	2	0
unknown a	play	2021	reg	rmse	1728	300000	200000	14	0	14		0

Table 2: **Metadata of Kaggle challenges.** Challenge types include "playground" (datasets from external sources or synthetically generated) and "featured" (datasets from real scientific or industrial applications, often with significant monetary prizes for top participants). Prediction tasks are binary classification (bin), regression (reg), or multi-class classification (mult; with the number of classes indicated in the #cls column). Note that in our method, *mult* and *bin* are treated the same. Features are categorized as numerical (num) or categorical (cat). The final column reports the number of missing entries in the training data.

B.2 Per-Challenge Results

Kaggle Challenge	**Meta-Informed**	**Zero-Shot**	**Context-Random**	**Random-Hyperopt**	**LGBM-Hyperopt**	**Context-Blends**
abalone	85.73 ± 3.3	74.67 ± 4.6	$\mathbf{87.87 \pm 2.3}$	58.95 ± 4.6	64.21 ± 11.3	92.06 ± 0.1
allstate	$\mathbf{69.92 \pm 2.3}$	61.66 ± 2.9	65.41 ± 5.0	50.05 ± 2.4	51.0 ± 2.7	77.15 ± 0.7
attrition	59.51 ± 1.7	$\mathbf{61.12 \pm 1.8}$	57.31 ± 2.3	59.36 ± 3.3	48.21 ± 5.0	57.47 ± 3.2
blueberry	$\mathbf{81.16 \pm 2.4}$	79.86 ± 1.7	78.96 ± 3.8	70.77 ± 5.3	65.87 ± 7.7	88.65 ± 0.8
churn	70.35 ± 0.9	68.73 ± 0.9	68.71 ± 3.0	65.07 ± 4.0	$\mathbf{70.64 \pm 1.0}$	71.48 ± 1.1
cirrhosis	70.58 ± 3.6	69.09 ± 1.4	$\mathbf{73.06 \pm 1.8}$	64.61 ± 4.6	70.17 ± 2.0	83.62 ± 2.7
concrete strength	74.34 ± 17.9	74.19 ± 6.8	59.37 ± 16.1	88.81 ± 5.4	$\mathbf{83.21 \pm 9.3}$	95.95 ± 2.8
covertype	$\mathbf{67.78 \pm 4.0}$	58.35 ± 7.6	60.05 ± 10.3	56.75 ± 11.0	32.0 ± 3.4	77.16 ± 1.0
crab age	$\mathbf{68.87 \pm 0.7}$	68.81 ± 0.6	67.67 ± 1.2	61.84 ± 2.3	63.84 ± 1.8	71.51 ± 0.2
credit fusion	96.61 ± 1.0	96.71 ± 1.1	90.91 ± 1.7	96.35 ± 0.9	$\mathbf{96.75 \pm 1.5}$	97.93 ± 0.8
failure	41.12 ± 1.5	43.52 ± 1.7	41.25 ± 0.8	43.7 ± 2.6	$\mathbf{48.15 \pm 7.0}$	38.87 ± 2.9
heat flux fi	$\mathbf{93.4 \pm 5.0}$	90.7 ± 4.3	83.65 ± 8.6	69.07 ± 6.6	36.22 ± 17.1	99.3 ± 0.1
horses	82.39 ± 7.7	$\mathbf{82.78 \pm 5.6}$	75.31 ± 10.6	81.15 ± 6.2	79.75 ± 5.7	73.73 ± 12.0
housing california	$\mathbf{62.53 \pm 0.6}$	54.84 ± 2.4	60.07 ± 2.0	46.9 ± 6.8	52.71 ± 3.9	71.57 ± 1.0
influencers	76.84 ± 7.4	83.55 ± 1.4	80.52 ± 2.8	82.95 ± 2.7	$\mathbf{87.45 \pm 1.9}$	74.24 ± 1.9
insurance	$\mathbf{74.68 \pm 2.4}$	68.16 ± 1.8	67.9 ± 2.1	62.53 ± 5.9	64.6 ± 3.4	84.46 ± 6.5
loan approval	71.58 ± 2.6	63.29 ± 5.5	66.84 ± 5.4	62.64 ± 6.9	$\mathbf{74.43 \pm 0.9}$	78.55 ± 0.9
media	$\mathbf{62.95 \pm 1.4}$	57.52 ± 2.0	61.81 ± 2.5	49.5 ± 7.5	26.07 ± 2.8	72.0 ± 0.6
mental health	$\mathbf{92.99 \pm 3.0}$	79.77 ± 10.2	89.69 ± 5.2	75.34 ± 9.5	80.11 ± 7.7	75.03 ± 5.2
mercedes	17.81 ± 2.8	36.44 ± 7.8	35.26 ± 10.6	$\mathbf{36.57 \pm 8.6}$	25.42 ± 2.0	59.43 ± 4.8
molecules	$\mathbf{97.52 \pm 1.5}$	96.34 ± 1.6	96.32 ± 3.3	96.33 ± 2.6	78.02 ± 12.6	83.63 ± 12.2
unknown a	$\mathbf{80.56 \pm 0.8}$	78.6 ± 0.8	72.59 ± 2.4	66.17 ± 2.5	61.41 ± 5.5	86.06 ± 1.4
Mean	$\mathbf{72.69 \pm 0.2}$	70.39 ± 0.2	70.02 ± 0.3	65.7 ± 1.1	61.8 ± 1.1	77.72 ± 0.2

Table 3: Kaggle p-rank results across all challenges (the higher, the better). Uncertainty is reported as $\pm$ values, representing the 90% confidence interval based on the standard error across 8 random seeds.

C Ensembling Pipeline

To evaluate and combine the model configurations proposed by the LLM, we implement a two stage ensembling pipeline using cross validation bagging (CV bagging) followed by feedforward greedy blending [Caruana et al., 2004].

CV-Bagging. Each base model is trained using k-fold cross-validation. For each fold, the model is trained on $k - 1$ partitions and evaluated on the held-out fold. This yields out-of-fold (OOF) predictions for the full training set, with no data leakage. These OOF predictions provide a reliable estimate of each model's generalization performance and serve as inputs to the blending stage.

Feedforward Greedy Blending. After collecting OOF predictions from all candidate models, we construct an ensemble using feedforward greedy blending. This method builds the blend iteratively: at each step, it adds the model that leads to the largest improvement on a validation score when combined (typically using a linear combination) with the current blend. The process continues until no further improvement is observed or a predefined limit on ensemble size is reached. Blending weights are determined incrementally during this selection process.

The final predictions on the test set are obtained by retraining each selected base model on the training data and applying the learned blend weights to their outputs. This Bag-Then-Blend pipeline is task-agnostic and metric-independent, making it suitable for systematic evaluation across diverse datasets and prediction objectives.

D Prompting Strategies

D.1 Current Task Description Format

For both prompting strategies, the LLM receives the current task description in the following structured format. Below is an example for the Abalone challenge:

```
# Metadata for kaggle_abalone

## name
kaggle_abalone

## prediction_type
regression

## score_name
rmsle

## n_train: 90615    n_test: 60411    total_samples: 151026    train_test_ratio: 1.5

## features
total: 9    numeric: 8    numerical_range_avg: 11327.82    categorical: 1

### unique_values_per_categorical
min: 3    max: 3    median: 3    mode: 3

## missing_data
has_missing: False    total_missing_values: 0    data_density: 1.0

## target_values
min: 1    max: 29    mean: 9.697    median: 9.0    std: 3.176    skewness: 1.204    kurtosis: 2.613
```

D.2 Zero-Shot Setting

The following system prompt is used for the Zero-Shot setting.

Zero-Shot System Prompt

> You are a data science expert specializing in model blending. You will receive a description of a machine learning tasks and dataset. Your task is to propose a new model blend with exactly 10 models by completing a given JSON file that describes a new task, maintaining the same format. You must output the json with 10 different choices of models and “models" as a key following exactly the input format JSON but removing the prank and mean score columns. Select models and hyperparameters considering factors such as dataset characteristics and task type. Don’t forget to give exactly 10 different variations and use the given format for the output adding the needed values lists. A predefined hyperparameter grid will be provided beforehand. Ensure your selections of the 10 models adhere to the available hyperparameter choices and that the number of models given is 10.

In the Zero-Shot setting, the LLM is not provided with in-context examples. To guide its output, it is instead given the expected JSON schema, as shown below.


```
{
  "models": {
    "catboost": {
      "columns": ["bootstrap_type", "border_count", "grow_policy", "l2_leaf_reg",
     "learning_rate",
      "max_depth", "min_data_in_leaf", "n_estimators", "random_strength"],
      "values": []
    },
    "lgbm": {
      "columns": ["boosting_type", "colsample_bynode", "colsample_bytree", "drop_rate",
      "learning_rate", "max_bin", "max_depth", "min_child_weight", "min_data_in_leaf",
      "min_split_gain", "n_estimators", "reg_alpha", "reg_lambda", "subsample"],
      "values": []
    },
    "xgboost": {
      "columns": ["colsample_bylevel", "colsample_bynode", "colsample_bytree", "gamma",
      "learning_rate", "max_depth", "min_child_weight", "n_estimators", "reg_alpha",
     "reg_lambda",
      "subsample"],
      "values": []
    },
    "skmlp": {
      "columns": ["activation", "alpha", "beta_1", "beta_2", "epsilon", "layers",
      "learning_rate_init", "max_iter", "n_iter_no_change", "n_knots"],
      "values": []
    }
  }
}
```

D.3 Meta-Informed Setting

The following system prompt is used for the Meta-Informed setting.

Meta-Informed System Prompt

You are a data science expert specializing in model blending. You will receive multiple descriptions of machine learning tasks, datasets, and the top 10 performing models in an blend in JSON format, including their individual mean scores and percentile ranks. Your task is to propose a new model blend with exactly 10 models by completing a given JSON file that describes a new task, maintaining the same format. You must output the json with 10 different choices of models and "models" as a key following exactly the input format JSON but removing the prank and mean score columns. Select models and hyperparameters based on the similarity between the new task and previous ones, considering factors such as dataset characteristics, task type, and model performance trends. Don't forget to give exactly 10 different variations and use the same format as the input. A predefined hyperparameter grid will be provided beforehand. Ensure your selections of the 10 models adhere to the available hyperparameter choices and that the number of models given is 10.

The prompt is enriched with information from other challenges, provided as pairs of task metadata and the top 10 models in the solution blend, formatted as previously described. For each given challenge, we include all other challenges of the same type (classification or regression).

E Chat API Configuration and Defaults

To invoke the DeepSeek-R1 API, we use the `client.chat.completions.create` function from the OpenAI SDK with default hyperparameters.

For more information, refer to the official documentation:

- OpenAI Platform: https://platform.openai.com/
- DeepSeek API Docs: https://api-docs.deepseek.com/

F Base Model Details

We use four base models in our experiments: XGBoost [Chen and Guestrin, 2016], CatBoost [Prokhorenkova et al., 2018], LGBM, and SKMLP [Pedregosa et al., 2011]. The corresponding hyperparameter grids for each model are provided in Figure 4.

CatBoost hyperparameter grid.

```
n_estimators = Hyperparameter(dtype='int', default=400, values=[10, 20, 30, 40, 50, 70, 100, 150, 200, 250, 300, 400, 500, 700, 1000])#, 2000, 3000, 5000, 7000, 10000])
learning_rate = Hyperparameter(dtype='float', default=0.05, values=[0.0005, 0.001, 0.002, 0.005, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
max_depth = Hyperparameter(dtype='int', default=5, values=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16])
l2_leaf_reg = Hyperparameter(dtype='float', default=3.0, values=[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 1.0, 2.0, 3.0, 4.0, 5.0])
border_count = Hyperparameter(dtype='int', default=254, values=[32, 64, 128, 254, 512, 1024])
grow_policy = Hyperparameter(dtype='str', default='SymmetricTree', values=['SymmetricTree', 'Depthwise', 'Lossguide'])
min_data_in_leaf = Hyperparameter(dtype='int', default=1, values=[1, 5, 10, 20, 50, 100, 200, 500, 700])
bootstrap_type = Hyperparameter(dtype='str', default='No', values=['No', 'Bernoulli', 'MVS', 'Bayesian_0', 'Bayesian_1', 'Bayesian_5', 'Bayesian_10', 'Bayesian_20',
'Bayesian_50'])
random_strength = Hyperparameter(dtype='float', default=1, values=[0, 1, 5, 10, 20, 50, 100])
```

LGBM hyperparameter grid.

```
colsample_bytree = Hyperparameter(dtype='float', default=0.5, values=[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
colsample_bynode = Hyperparameter(dtype='float', default=0.5, values=[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
min_split_gain = Hyperparameter(dtype='float', default=0.0, values=[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.2, 1.5, 2.0])
learning_rate = Hyperparameter(dtype='float', default=0.05, values=[0.0005, 0.001, 0.002, 0.005, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
max_depth = Hyperparameter(dtype='int', default=5, values=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20, 25, 30, 35, 40, 50, 60, 70, 80, 90, 100])
min_child_weight = Hyperparameter(dtype='int', default=9, values=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
n_estimators = Hyperparameter(dtype='int', default=400, values=[10, 20, 30, 40, 50, 70, 100, 150, 200, 250, 300, 400, 500, 700, 1000])#, 2000, 3000, 5000, 7000, 10000])
reg_alpha = Hyperparameter(dtype='float', default=2.0, values=[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 1.0, 2.0, 3.0])
reg_lambda = Hyperparameter(dtype='float', default=5.0, values=[0.5, 1.0, 1.5, 2.0, 3.0, 4.0, 5.0])
subsample = Hyperparameter(dtype='float', default=0.9, values=[0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
max_bin = Hyperparameter(dtype='int', default=256, values=[256, 512, 1024, 2048, 4096, 8192])
min_data_in_leaf = Hyperparameter(dtype='int', default=1, values=[1, 5, 10, 20, 50, 100, 200, 500, 700])
boosting_type = Hyperparameter(dtype='str', default='gbdt_5', values=['gbdt_0', 'gbdt_1', 'gbdt_5', 'gbdt_10', 'dart_0', 'dart_1', 'dart_5', 'dart_10', 'goss'])
drop_rate = Hyperparameter(dtype='float', default=0.1, values=[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
```

XGBoost hyperparameter grid.

```
colsample_bytree = Hyperparameter(dtype='float', default=0.5, values=[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
colsample_bylevel = Hyperparameter(dtype='float', default=0.5, values=[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
colsample_bynode = Hyperparameter(dtype='float', default=0.5, values=[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
gamma = Hyperparameter(dtype='float', default=0.0, values=[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.2, 1.5, 2.0])
learning_rate = Hyperparameter(dtype='float', default=0.1, values=[0.0005, 0.001, 0.002, 0.005, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
max_depth = Hyperparameter(dtype='int', default=2, values=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20, 25, 30, 35, 40, 50, 60, 70, 80, 90, 100])
min_child_weight = Hyperparameter(dtype='int', default=1, values=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
n_estimators = Hyperparameter(dtype='int', default=700, values=[10, 20, 30, 40, 50, 70, 100, 150, 200, 250, 300, 400, 500, 700, 1000])#, 2000, 3000, 5000, 7000, 10000])
reg_alpha = Hyperparameter(dtype='float', default=0.1, values=[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 1.0, 2.0, 3.0])
reg_lambda = Hyperparameter(dtype='float', default=0.5, values=[0.5, 1.0, 1.5, 2.0, 3.0, 4.0, 5.0])
subsample = Hyperparameter(dtype='float', default=1.0, values=[0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
```

SKMLP hyperparameter grid.

```
layers = Hyperparameter(dtype='str', default='512-256', values=['32', '64', '128', '256', '256-128', '512-256'])#, '1024-512', '1024-512-512'])
activation = Hyperparameter(dtype='str', default='tanh', values=['relu', 'tanh', 'logistic'])
alpha = Hyperparameter(dtype='float', default=0.1, values=[0.0001, 0.001, 0.01, 0.1])
learning_rate_init = Hyperparameter(dtype='float', default=0.01, values=[0.001, 0.01, 0.1])
max_iter = Hyperparameter(dtype='int', default=5000, values=[5000, 10000, 20000])
n_iter_no_change = Hyperparameter(dtype='int', default=10, values=[5, 10, 20])
beta_1 = Hyperparameter(dtype='float', default=0.8, values=[0.8, 0.9, 0.95])
beta_2 = Hyperparameter(dtype='float', default=0.999, values=[0.99, 0.999, 0.9999])
epsilon = Hyperparameter(dtype='float', default=1e-7, values=[1e-8, 1e-7, 1e-6])
n_knots = Hyperparameter(dtype='int', default=5, values=[3, 5, 10, 20])
```

Figure 4: Base models hyperparameters.

G Example Reasoning Traces

To illustrate the model's decision-making process, we show a few selected excerpts from its reasoning traces on different Kaggle tasks.

Mercedes Dataset (Zero-Shot)

Okay, so I need to create a new model blend for the `Kaggle Mercedes` regression task. The dataset has 377 features, mostly categorical (376) and one numeric. The target is continuous with some skewness. The goal is to propose 10 different models across CatBoost, LGBM, XGBoost, and SKMLP, using the given hyperparameters.
First, let's think about the dataset characteristics. Since there are a lot of categorical features, tree-based models like CatBoost, LGBM, and XGBoost might perform well because they handle categorical variables effectively. `skmlp` might need more tuning but could add diversity to the blend.
For CatBoost, since it's good with categorical data, I'll focus on parameters that handle that. `bootstrap_type` could be Bayesian variants for better handling. `border_count` might be higher (like 512 or 1024) to capture more splits. `grow_policy` could be `Lossguide` for deeper trees. `learning_rate` should be moderate, maybe 0.05 to 0.1. L2 regularization (`l2_leaf_reg`) around 3.0 to prevent overfitting. `n_estimators` around 500–700 for sufficient depth. `random_strength` maybe 1.0 or 5.0 for some randomness.
For LGBM, `boosting_type` like `goss` or `dart` could be efficient. Since there are many features, `colsample_bytree` and `subsample` around 0.8 to reduce overfitting. `learning_rate` similar to CatBoost. `max_depth` around 7–10. `min_data_in_leaf` higher (like 50) given the dataset size. `n_estimators` around 300–500. Regularization parameters (`reg_alpha` and `reg_lambda`) around 0.5–2.0.
XGBoost would need similar considerations. `colsample_bytree` and `subsample` around 0.8. `learning_rate` 0.05–0.1. `max_depth` 6–10. `gamma` around 0.5 to control splits. `reg_alpha` and `reg_lambda` similar to LGBM. `n_estimators` 300–500.
`skmlp` might have layers like `512-256` to handle high dimensionality. `activation=relu` for non-linearity. `learning_rate_init=0.01`. `alpha` (L2 regularization) 0.001 or 0.01. Beta parameters (Adam optimizer) default to 0.9 and 0.999. `max_iter` high (20000) since MLPs need time. `n_iter_no_change=10`. `n_knots` maybe 10 for spline layers.
I need to ensure 10 models in total. Maybe distribute them as 3 CatBoost, 3 LGBM, 3 XGBoost, and 1 SKMLP for diversity. Each model should have unique hyperparameters within the grid. Check each parameter against the allowed values. For example, CatBoost's `bootstrap_type` can't be "No" but has Bayesian options. Ensure all selected values are in the provided grid. Also, no duplicates in the models. Each entry must have distinct hyperparameter combinations.

Abalone Dataset (Meta-Informed)

Okay, I need to create a new model blend for the `kaggle_abalone` task. Let me start by understanding the problem. The task is regression with RMSLE as the score. The dataset has 9 features, 8 numeric and 1 categorical with 3 unique values. The target has a mean around 9.7, with some skewness and kurtosis. The data is clean with no missing values.
Looking at the past tasks, `kaggle_crab_age` seems similar. Both have regression, a categorical feature with 3 unique values, and similar target distributions (mean around 9–10, skewness > 1). In `kaggle_crab_age`, the top models were CatBoost and LGBM. For example, CatBoost used `bootstrap_type=Bernoulli`, high `border_count=1024`, `grow_policy=Lossguide`, and `n_estimators` around 500–700. LGBM models had `boosting_type=gbdt_1` or `gbdt_5`, high `n_estimators` (1000), and specific regularization parameters.
Also, `kaggle_unknown_a` had a similar setup with all numeric features and high `n_estimators`. XGBoost models there used `colsample_bytree` around 0.5–0.8 and `learning_rate` 0.01–0.1. SKMLP models used `activation=logistic`, low `alpha=0.0001`, and layers such as `256-128`.
Considering hyperparameters from the grid: for CatBoost, possible `bootstrap_type` values include Bernoulli or Bayesian, `border_count=1024`, `grow_policy=Lossguide` or `Depthwise`, `l2_leaf_reg` around 0.2–3.0, `learning_rate=0.1`, `max_depth` around 9–12, and `min_data_in_leaf=700`. For LGBM, suitable choices are `boosting_type=gbdt_1` or `gbdt_5`, `colsample_bynode` 0.8–1.0, `learning_rate=0.05`, `max_depth` around 90, and `min_data_in_leaf=50`. XGBoost could use `colsample_bytree` 0.5–0.8, `gamma` 0.0–1.5, `learning_rate` 0.05–0.1, `max_depth=9`, and `n_estimators=1000`. For SKMLP, I will use `activation=logistic`, `alpha=0.0001`, layers `256-128`, and `learning_rate_init=0.001`.
I need to select 10 models across different algorithms. Let's pick 4 CatBoost, 3 LGBM, 2 XGBoost, and 1 SKMLP, ensuring hyperparameters are drawn from the predefined grid. Check each parameter against the allowed values (e.g., CatBoost `bootstrap_type`, LGBM `boosting_type`), and ensure `n_estimators` is valid. Each model must be unique in its parameter combination. Finally, I will compile the JSON structure without `mean_score` and `prank`, keeping the same format as the input examples.

Heat Flux Fi Dataset (Meta-Informed)

Okay, I need to create a new model blend for the `kaggle_heat_flux_fi` task. Let's start by understanding the new task's characteristics. The prediction type is regression, and the score is RMSE. The dataset has 21,229 training samples and 10,415 test samples. There are 9 features: 7 numeric and 2 categorical. The numeric features have an average range of about 6,121, which is similar to some previous tasks like `kaggle_abalone` or `kaggle_crab_age`. The categorical features have 4 and 11 unique values, which is a moderate number. There's missing data with a data density of 0.88, which means some imputation might be needed. The target has a mean close to zero, a negative skewness, and high kurtosis, indicating a peaky distribution with a left tail.

Looking at past tasks, models that performed well on datasets with mixed numeric and categorical features, especially those using CatBoost and LGBM, might be relevant. For example, `kaggle_abalone` used CatBoost with `grow_policy=Lossguide` and high `border_count`, which handles categorical features well. Also, tasks with missing data like `kaggle_media` had LGBM models with certain parameters.

The hyperparameter grids need to be considered. For CatBoost, handling missing data is inherent, so models with `bootstrap_type` like Bayesian or MVS might be good. LGBM can handle missing values too. XGBoost doesn't handle categoricals natively, so maybe less emphasis here unless the categoricals are encoded. Since the new task has categorical features, CatBoost and LGBM are probably better choices.

Looking at the previous top models, for regression tasks with similar features, CatBoost models with higher `border_count` (like 1024) and LGBM with higher `max_depth` (like 40, 90) and `n_estimators` (1000) performed well. Also, models with lower learning rates (0.1, 0.15) and regularization (`l2_leaf_reg` around 0.5–2.0) were common.

Considering the new task's missing data, models that handle missing values well are crucial. CatBoost's `bootstrap_type` options like Bayesian or MVS, and LGBM with `drop_rate` and `subsample` parameters could be beneficial. Also, since the target has a negative skew, maybe models that are robust to such distributions, like those with appropriate loss functions or regularization.

Now, selecting 10 models. Let's check the hyperparameter grids to ensure valid choices. For CatBoost, possible `bootstrap_type`: Bayesian_0, MVS, Bernoulli. `border_count` options: 32, 64, 128, etc. For LGBM, `boosting_type`: `gbdt_0`, `dart_1`, `goss`. Also, `subsample` and `colsample` parameters. `skmlp` might be less prioritized due to the categorical features, but maybe include one if previous tasks with similar data used it.

Looking at the previous models, in `kaggle_unknown_a`, CatBoost with `bootstrap_type=No` and `grow_policy=Depthwise` was used. For the new task, maybe mix different bootstrap types. For LGBM, models with higher `n_estimators` (1000) and `max_depth` (like 40, 90) from `kaggle_abalone` and `kaggle_housing_california`.

Also, in the new task, the data has missing values, so models that handle missing data without imputation (like CatBoost and LGBM) are preferred. XGBoost requires handling missing data externally, so maybe less emphasis unless parameters are adjusted, but the hyperparameter grid doesn't include handling for missing data, so maybe avoid XGBoost unless necessary.

Looking at `skmlp`, in past tasks like `kaggle_insurance`, `activation=logistic` with certain layers was used. Maybe include one `skmlp` model for diversity.

Now, compiling 10 models.

H Baselines description

H.1 Context-Random

For the **Context-Random** baseline, we uniformly sample k model-hyperparameter configurations from the same pool of prior-task blends that are provided as context in the **Meta-Informed** setting. This isolates whether improvements come from meaningful adaptation by the LLM or simply from re-using high-quality configurations already present in the context.

We fix $k = 10$ to match the number of configurations proposed by the LLM in a single run.

H.2 Random-Hyperopt

For the **Random-Hyperopt** baseline, we use HEBO to optimize hyperparameters within a model family, but the model family itself is selected uniformly at random at each round. Concretely, at each iteration one of the base learners is sampled with equal probability, after which HEBO proposes a new configuration for that family. This ensures a simple exploration strategy without bias toward any particular model type.

H.3 LGBM-Hyperopt

For the **LGBM-Hyperopt** baseline, we restrict the search space to the LightGBM model family. At each evaluation round, we apply the HEBO optimizer to propose a new LightGBM configuration, which is then trained and evaluated on the target dataset. This baseline isolates the performance of hyperparameter optimization when applied to a single strong gradient boosting method without model family selection. As with the other baselines, we allocate a fixed budget of 10 evaluations when comparing against the LLM recommendations.

H.4 MaxUCB-Hyperopt

For the **MaxUCB-Hyperopt** baseline, we implement the bandit-based CASH formulation proposed by Balef et al. [2025]. In this setting, each candidate model family is treated as an arm in a multi-armed bandit, and hyperparameter optimization is carried out within the selected arm using HEBO. The Max-UCB algorithm balances exploration of new model families with exploitation of those that have already demonstrated promising performance.

At each round t, the utility of arm i is computed as:

$$U_i = \max(r_{i,1}, \ldots, r_{i,n_i}) + \left(\frac{\alpha \log(t)}{n_i} \right)^2,$$

where $r_{i,j}$ denotes the observed rewards (validation scores) from the j-th configuration of model family i, and n_i is the number of configurations tried so far for that family. The algorithm selects the arm

$$I_t = \arg\max_{i \leq K} U_i,$$

applies HEBO within that model family to propose a new hyperparameter configuration, and observes the resulting reward.

Following recommendations from the original paper, we set the exploration parameter to $\alpha = 0.5$, which provides a favorable balance between exploration and exploitation across tasks.

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 abstract and introduction claim (i) LLMs can act as in-context meta-learners for CASH, (ii) Meta-Informed prompting outperforms Zero-Shot and Hyperopt baselines on 22 Kaggle tasks, and offer interpretability/efficiency. these match the results given in (Figs. 2, 3 and Table 1).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: While we do not have a separate limitations section, we note several limitations such as the fact that performance is still below expert AutoML blends as well as variable gains across datasets with some tasks favoring baselines discussed in Subsection 2.3.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

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: No theoretical result.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

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: We specify in the Appendix and Section 2 datasets and model grids, prompting strategies, LLM and API setting, baseline details and the ensembling pipeline.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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: [No]

   Justification: We do not release code in the submission.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

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: We specify in the Appendix and Section 2 datasets and model grids, prompting strategies, LLM and API setting, baseline details and the ensembling pipeline.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

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: We report 90% confidence intervals via standard error across for plots and tables (Fig. 2, Fig. 3, Table 1).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

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: [No]

   Justification: We fix a round budget and number of models (Section 2) but do not enumerate hardware specs, memory, or wall-clock times.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

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: The work uses public/tabular datasets and standard AutoML components, involves no human subjects or sensitive personal data beyond what is present in the public benchmarks, and cites all sources

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [NA]

    Justification: No societal impact

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

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: No specific risk

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

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 papers and competitions/datasets with their url. Licence and terms of use are not explicitly mentioned.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are used in the method. The way they are used is described in the introduction and Section 2.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.