Semantic feature engineering for in-context AutoML

Afonso Lourenço[1,2,*] Hiroaki Kingetsu[2] Tsuguchika Tabaru[2] Goreti Marreiros[1]
[1] GECAD, Polytechnic of Porto, Portugal [2] Fujitsu Laboratory Ltd., Japan
[*] Internship at Fujitsu
{fonso,mgt}@isep.ipp.pt {h.kingetsu,tabaru}@fujitsu.com

Abstract

Machine learning for structured data has lagged behind text and image, with current methods remaining application-dependent and requiring extensive algorithm selection and hyperparameter tuning. Large tabular models (LTMs) offer a promising solution for context-aware AutoML by pretraining on diverse tabular datasets. However, scalability remains a challenge due to the quadratic growth of contexts. This paper introduces a novel in-context AutoML paradigm focused on semantically informed feature engineering, where input data, rather than model parameters, are treated as learnable components. By leveraging task-specific insights from data card descriptions and historical logs, a large language model (LLM) enhances context creation for a LTM. Empirical results on ten benchmark datasets demonstrate this paradigm delivers competitive performance compared to conventional AutoML methods.

1 Introduction

Over the past two decades, machine learning for structured data has lagged behind advances in text and image modalities. Benchmark studies still find gradient-boosted decision trees as the state-of-the-art for supervised tabular learning [1]. The absence of spatial invariances to inform prior selection as well as discontinuous, heterogeneous, and uninformative features make the selection of a machine machine learning pipeline highly application-dependent, covering two critical stages: data pre-processing (wrangling, integration, and transformation to improve feature quality) and model building (automating algorithm selection and hyperparameter tuning), shown in Figure 1.

Despite traditional AutoML methods simplifying the process, these typically search iteratively from scratch for new tasks, ignoring human priors and historical insights, such as model architecture knowledge. This results in time-consuming and computationally expensive cycles of model selection and hyperparameter tuning. Additionally, these methods rely on predefined strategies, limiting their ability to adapt based on insights from the learning process, and their black-box nature further reduces interpretability, excluding human understanding from the process [2, 3, 4].

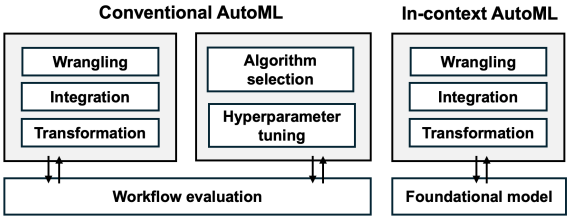

Figure 1 New AutoML paradigm driven by LTMs

To tackle these issues, one should leverage both machine intelligence and human design patterns. Instead of jumping into solving a new task directly, one should utilize all contextual information to interpret the task at hand, and draw from past experiences. In this regard, recently proposed large tabular models (LTMs) offer a natural solution to context-aware AutoML, by finetuning a cross-table pretrained tabular transformer, instead of training models on single tables [5]. Similarly to large language models (LLMs), LTMs trained on diverse tabular datasets can serve as knowledge repositories [6, 7]. These obviate the need for algorithm selection and hyperparameter tuning, with the bottleneck lying in contexts scaling quadratically. Naturally, this limits performance, and contrasts with coventional AutoML that tends to improve as the amount of data increases. Thus, posing a new in-context paradigm, referred as IC-AutoML and shown in Figure 1, solely focused on data pre-processing to determine what to actually put in a pre-tuned model’s context. While conventional methods need to adapt the model parameters, this paradigm focuses on input data as learnable parameters, decoupling algorithm design from a general-purpose LTM [8].

To determine the model's context, various strategies for transforming a big dataset into a sketch have been proposed [9]. Yet these are exclusively data-driven, not understanding the task from a semantic perspective. Instead, this work hypothesizes that crucial information to classify a query point can be found semantically via LLMs, incorporating descriptions of data cards, and historical logs into prompts, to not only engineer compact feature representations but also explain the choices behind a pre-processing operation for context optimization. For example, understanding the relationship between a city and its zip code for data cleaning constraints, or how age is intuitively more relevant than ward census for disease prediction [10]. Similarly to how recent works have used LLMs for optimization of another LLM [11], this work uses LLMs for optimization of a LTM, acting as interactive agents, capable of generating code and prompt editing. Experiments performed on ten benchmark datasets show that the new proposed paradigm is as good as current state of the art methods for tabular classification, i.e. XGBoost [12], LightGBM [13], CatBoost [14], AutoGluon [2], H_2O [3], and PyCaret [4].

2 Related work

Recent efforts have explored applying LLMs' validated capabilities in table understanding to supervised learning of tabular data, predicting unseen samples by continuing textual descriptions, e.g., "The column name is Value" [15]. When supported with extensive tabular-specific pretraining, tabular prompting in LLMs can outperform traditional methods [16]. Nonetheless, tokenization and multinomial objectives of LLMs often result in fragmented patterns, making it expensive to autoregressively model continuous variables or whole numbers. To overcome this, recent efforts have introduced LTMs, i.e. transformer-based architectures which are specifically trained on heterogeneous tabular datasets to better handle numeric data [6]. For instance, allowing to represent a decimal feature with a single floating-point number, whereas LLAMA requires four tokens for the same value [17]. However, LTM's quadratic complexity cannot scale with samples, features, and categories, which limits its application to real-world scenarios. Thus, similarly to LLM prompt engineering [18], various context optimization techniques have been proposed, including k-means centroids [19], dataset distillation [20], and retrieval-based strategies [21].

These data-driven context optimization methods lack semantic task understanding. In this context, LLMs can offer significant promise as black-box optimizers to iteratively generate decision rules [22], and code for feature engineering (FE) using user-provided dataset descriptions, feature names, data types, missing values, and random samples [23], with iterative feature selection guided by validation accuracy scores [24, 23]. Building on previous work that combines LLMs with genetic algorithm (GA) optimization for tasks like neural architecture search [25], and feature selection [24], this paper proposes using LLMs as evolutionary operators within a GA framework to optimize FE for LTM context optimization.

3 Methodology

In this work, in-context Auto-ML is referred to as the construction of a map from in-context examples to a LTM without any updates to the LTM's parameters. From a variance standpoint, being a pre-tuned, but untrained predictor with many hyperparameters and multi-head attention, LTMs have extremely high sensitivity to individual training samples, which translates in an increased ability to choose submodels and vanishing variance. From a bias standpoint, hyperparameters are pre-tuned to be optimal for a set of tasks defined by the prior [8]. If the prior has broad support and does not concentrate away from the true hypothesis, the posterior predictive distribution (PPD) approximates the true distribution. Larger datasets lead to more complex PPDs, and the training set size acts as a regularizer on the model's complexity. While LTMs ensure vanishing variance, bias decreases only if the context is appropriately localized around the test feature. The in-context learning error decreases with a more informative input space [26], underscoring the need for localization strategies that restrict the context to concept-related examples. This highlights the role of inductive biases, such as smoothness, cluster, and manifold assumptions, in guiding model adaptation. Leveraging human knowledge to bias the parameter space structure can yield state-of-the-art AutoML, enhancing sample efficiency and generalization.

Building on this intuition, this work proposes a localization strategy based on semantic similarity to reduce model bias. Particularly, one can argue that the context is not only sensitive to the choice of the instances but also the input features used to represent data, suggesting that improvements

in the earlier context optimization work [19, 20, 21] may be possible. To address this, one can use LLMs proven capabilities on FE [23, 24] to significantly enrich the engineered context, with devised prompts including details about the dataset's collection, task's objective, explanation of the target variable, feature descriptions, and samples of instances values. For this purpose, this work relies on various prompt engineering techniques for multi-step reasoning, namely, chain-of-thought (CoT) [18], role-playing [27], and other tabular specific serialization techniques [15]. Moreover, it is important to emphasize both exploration and exploitation of the prompt space. On one hand, to deal with the high variance of different contexts for the downstream tasks, one should explore the space by enumerating and selecting the best prompt from a number of candidates, e.g. augmenting it by re-sampling [28]. On the other hand, to emphasize exploitation, one can collect the incorrectly predicted cases and analyze the corresponding root cause to edit existing prompts [11]. Building on previous work that combines LLMs with GA optimization [24, 25], this work uses LLMs as a evolutionary operator within a GA framework to efficiently explore and exploit the FE solution space. Overall, this GA-inspired optimization framework encompasses six steps: (1) population initialization derived by prompting the LLM Y times in a zero-shot manner, with task description, features, and data types, (2) tournament selection method of parents for evolutionary pressure to the optimization process, (3) LLM as the evolutionary operator via few-shot role-playing and chain-of-tought explanations, (4) child evaluation via LLM-generated code to run the LTM with the suggested features, (5) elitism replacement based on the obtained validation accuracy, with a hard constraint of not replacing the top K feature combinations, (6) random immigrant invoked when the best solution does not change for R epochs, by randomly replacing a solution with validation accuracy under the mean by a new solution generated via zero-shot prompting. Prompt templates can be found in the Appendix. This iterative, evolutionary process, is repeated for E epochs, culminating in the selection of the best one within the final population as the optimal data pre-processing strategy.

4 Experiments

In these experiments, LLaMA3 70B [17] was used as the backbone LLM, and the TabPFN [6] as the LTM. The initialization stage begins with the LLM being asked to assume the role of a Machine Learning Engineer to recommend a list of important features based on the given task and dataset features in a zero-shot manner. The LLM is prompted twenty times to generate different feature sets, forming the initial population for the evolution algorithm. The evolution stage comprises 17 distinct roleplays: Domain Expert, Public Policy, Philosopher, Consultant, Professor, Coach, Data Scientist, Data Analyst, Machine Learning Engineer, Manager, AI/ML Researcher, Data Engineer, Ethical AI Advocate. These roleplays facilitate crossover and mutation operations within the LLM, leveraging diverse perspectives for semantically-driven feature engineering. While the effectiveness of this approach depends heavily on the quality of dataset descriptions, prior studies [22, 23, 24] have explored these effects in detail. Building on their findings, we included in the prompts a detailed dataset context, input and target concepts, few-shot examples, and CoT explanations. To ensure comprehensive evaluation, a diverse set of binary classification datasets were used: Credit[1)] (150.000 instances, 11 features, imbalanced), Diabetes[2)] (70.692 inst., 22 feat., bal.), Heart[3)] (70.000 inst., 12 feat., bal.), Insurance[4)] (50.882 inst., 48 feat., imb.), Bank[5)] (45.211 inst., 38 feat., imb.), Cars[6)] (16.734 inst., 42 feat., bal.), Stroke[7)] (9.722 inst., 22 feat., bal.), Student[8)] (4.424 inst., 57 feat., imb.), Credit-G[9)] (1.000 inst., 17 feat., bal.), and Pima Indians Diabetes[10)] (536 inst., 8 feat., bal.).

These datasets were selected for their suitability in evaluating the impact of semantically-driven feature engineering on model performance, as they present problems amenable to domain knowledge-based feature representations. Table 1 summarizes accuracy performance across these, ordered by dataset size and evaluated over ten seeds. Hyperparameter settings can be found in the Appendix. IC-AutoML demonstrated competitive results, performing best on the

1) kaggle.com/c/GiveMeSomeCredit
2) kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset
3) kaggle.com/datasets/sulianova/cardiovascular-disease-dataset
4) kaggle.com/datasets/owaiskhan9654/health-insurance-lead-prediction-raw-data
5) archive.ics.uci.edu/dataset/222
6) kaggle.com/datasets/nelgiriyewithana/australian-vehicle-prices
7) kaggle.com/datasets/fedesoriano/stroke-prediction-dataset
8) archive.ics.uci.edu/dataset/697
9) kaggle.com/datasets/uciml/german-credit
10) kaggle.com/datasets/uciml/pima-indians-diabetes-database

Table 1 Accuracy performance

	IC-AutoML	XGBoost	LightGBM	CatBoost	AutoGluon	H_2O	PyCaret
Credit	**.730 ± .022**	.662 ± .030	.658 ± .031	.688 ± .027	.692 ± .026	.722 ± .023	.690 ± .028
Diabetes	.758 ± .029	.764 ± .018	**.774 ± .017**	.758 ± .020	.772 ± .019	.736 ± .022	.768 ± .018
Heart	**.766 ± .017**	.758 ± .019	**.766 ± .016**	**.766 ± .018**	.758 ± .021	.748 ± .020	.756 ± .019
Insurance	.726 ± .020	**.730 ± .022**	.702 ± .025	.710 ± .023	.708 ± .021	.724 ± .020	.702 ± .024
Bank	.828 ± .027	.745 ± .019	.792 ± .016	.778 ± .021	.814 ± .015	**.878 ± .013**	.844 ± .016
Cars	**.836 ± .038**	.788 ± .019	.796 ± .018	.794 ± .019	.818 ± .016	.824 ± .014	.812 ± .017
Stroke	.930 ± .045	.946 ± .016	.952 ± .014	.950 ± .015	**.962 ± .010**	.958 ± .012	.950 ± .013
Student	.910 ± .034	.864 ± .022	.850 ± .021	.890 ± .015	.886 ± .018	**.914 ± .012**	.892 ± .015
Credit-g	**.754 ± .043**	.712 ± .027	.693 ± .033	.701 ± .029	.740 ± .024	.748 ± .022	.736 ± .025
Pima	**.794 ± .017**	.786 ± .019	.778 ± .020	.783 ± .018	.767 ± .022	.791 ± .016	.788 ± .017
Avg. Rank	**2.55**	5.05	4.95	4.80	3.75	2.90	4.0

Credit (.730), Cars (.836), Credit-g (.754) and Pima (.794) datasets. Overall, achieving an average rank of 2.55 across diverse datasets, showcasing its general effectiveness. Among other methods, H_2O achieved the highest average rank (2.90), particularly excelling on the Bank (0.878) and Student (0.914) datasets. XGBoost, LightGBM, and CatBoost performed best on larger datasets like Diabetes (.774), Heart (.766), and Insurance (.730), with average ranks around 5.0. Interestingly, IC-AutoML did not show a clear advantage on smaller datasets, despite processing contexts limited to 1,000 instances at a time.

The engineered features varied widely across datasets. For example, in the Credit dataset, the LLM proposed 47 new features, with the final selection comprising 13 of them, including log transformations to normalize skewed features such as revolving utilization and debt ratio, binning operations to flag high debt-to-income ratios, and interaction features like income per dependent or a composite delinquency score. In healthcare datasets like Diabetes (42 features suggested), Heart (43), and Stroke (56), domain knowledge-driven feature interactions proved most beneficial. For instance, in the Stroke dataset, the LLM suggested a stress proxy combining employment type, urban residence, smoking status, and marital history; a health deterioration rate integrating glucose levels, BMI, and age; and a loneliness index derived from marital status, employment history, and age. Conversely, in the Diabetes dataset, suggested features included a socioeconomic disparity indicator combining income and education levels, a proactive health behavior score averaging cholesterol checks, blood pressure management, and healthcare coverage, and a comorbidity score summing key health risk factors (e.g., high blood pressure, smoking). Another impactful feature was a health risk-adjusted age metric, calculated as the product of age and a composite health risk score.

5 Conclusion

This work shows how LTMs enable a promising learning paradigm, referred to as IC-AutoML. Rather than relying on extensive model selection and hyperparameter tuning, this approach focuses on data pre-processing as a localization strategy for a pre-tuned, yet untrained, LTM. Specifically, this work employs LLMs as evolutionary operators within a GA framework to explore the feature engineering solution space, utilizing prompts enriched with dataset details, task objectives, target variable explanations, feature descriptions, and instance samples. Dynamic role selection [27] expands the configuration space, potentially improving solutions at the cost of increased computation. To address this, prior evaluations of feature engineering decisions refine the configuration process CoT [18].

This work emphasizes the context sensitivity to input features, leveraging semantically-driven FE. Future research will explore other pre-processing operations, such as error detection and data cleaning [29], along with the optimization of instance selection [19, 20, 21]. Additionally, the interaction with users in a human-in-the-loop IC-AutoML framework warrants further investigation.

Acknowledgements

This work is supported by Fujitsu Laboratory Ltd. and FCT under project doi.org/10.54499/UIDP/00760/2020.

References

[1] Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? **Advances in Neural Information Processing Systems**, Vol. 36, , 2024.

[2] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. **arXiv preprint arXiv:2003.06505**, 2020.

[3] P. Stetsenko. **Machine Learning with Python and H2O**, October 2022.

[4] Moez Ali. **PyCaret: An open source, low-code machine learning library in Python**, April 2020.

[5] Bingzhao Zhu, Xingjian Shi, Nick Erickson, Mu Li, George Karypis, and Mahsa Shoaran. Xtab: Cross-table pretraining for tabular transformers. **arXiv preprint arXiv:2305.06090**, 2023.

[6] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. **arXiv preprint arXiv:2207.01848**, 2022.

[7] David Bonet, Daniel Mas Montserrat, Xavier Giró-i Nieto, and Alexander G Ioannidis. Hyperfast: Instant classification for tabular data. In **Proceedings of the AAAI Conference on Artificial Intelligence**, Vol. 38, pp. 11114–11123, 2024.

[8] Thomas Nagler. Statistical foundations of prior-data fitted networks. In **International Conference on Machine Learning**, pp. 25660–25676. PMLR, 2023.

[9] Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. A survey on data selection for language models. **arXiv preprint arXiv:2402.16827**, 2024.

[10] Zifeng Wang, Chufan Gao, Cao Xiao, and Jimeng Sun. Meditab: Scaling medical tabular data predictors via data consolidation, enrichment, and refinement. **arXiv preprint arXiv:2305.12081**, 2023.

[11] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with" gradient descent" and beam search. **arXiv preprint arXiv:2305.03495**, 2023.

[12] T Chen. Xgboost: extreme gradient boosting. **R package version 0.4-2**, Vol. 1, No. 4, 2015.

[13] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. **Advances in neural information processing systems**, Vol. 30, , 2017.

[14] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. **Advances in neural information processing systems**, Vol. 31, , 2018.

[15] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In **International Conference on Artificial Intelligence and Statistics**, pp. 5549–5581. PMLR, 2023.

[16] Josh Gardner, Juan C Perdomo, and Ludwig Schmidt. Large scale transfer learning for tabular data via language modeling. **arXiv preprint arXiv:2406.12031**, 2024.

[17] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. **arXiv preprint arXiv:2302.13971**, 2023.

[18] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. **Advances in neural information processing systems**, Vol. 35, pp. 24824–24837, 2022.

[19] Benjamin Feuer, Robin Tibor Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. Tunetables: Context optimization for scalable prior-data fitted networks. **arXiv preprint arXiv:2402.11137**, 2024.

[20] Junwei Ma, Valentin Thomas, Guangwei Yu, and Anthony Caterini. In-context data distillation with tabpfn. **arXiv preprint arXiv:2402.06971**, 2024.

[21] Andreas C Mueller, Carlo A Curino, and Raghu Ramakrishnan. Mothernet: Fast training and inference via hypernetwork transformers. In **NeurIPS 2024 Third Table Representation Learning Workshop**.

[22] Sungwon Han, Jinsung Yoon, Sercan O Arik, and Tomas Pfister. Large language models can automatically engineer features for few-shot tabular learning. **arXiv preprint arXiv:2404.09491**, 2024.

[23] Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. **Advances in Neural Information Processing Systems**, Vol. 36, , 2024.

[24] Shaoshan Liu, Fuyuan Lvu, Xue Liu, et al. Ice-search: A language model-driven feature selection approach. **arXiv preprint arXiv:2402.18609**, 2024.

[25] Angelica Chen, David Dohan, and David So. Evoprompting: language models for code-level neural architecture search. **Advances in Neural Information Processing Systems**, Vol. 36, , 2024.

[26] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. **arXiv preprint arXiv:2111.02080**, 2022.

[27] Murray Shanahan, Kyle McDonell, and Laria Reynolds. Role play with large language models. **Nature**, Vol. 623, No. 7987, pp. 493–498, 2023.

[28] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. **arXiv preprint arXiv:2211.01910**, 2022.

[29] Avanika Narayan, Ines Chami, Laurel Orr, Simran Arora, and Christopher Ré. Can foundation models wrangle your data? **arXiv preprint arXiv:2205.09911**, 2022.

A Appendix

Detailed prompt examples used for various tasks are provided. Note that text outputs from the language model require further processing to extract relevant information, yet, for simplicity, these are omitted in the following prompts.

A.1 Zero-shot initialization

Enhance predictions for {Objective} using a dataset: {Method of collection}.
Available attributes:

- {Column A Name}: (Numerical, range: {Min}–{Max}), Samples: [...], {Description}
- {Column B Name}: (Boolean, {0 = X}, {1 = Y}), Samples: [...], {Description}
- {Column C Name}: (Categorical, {1 = X}, {2 = Y}, {3 = Z}), Samples: [...], {Description}

Propose new feature combinations to improve prediction performance using transformations, interactions, aggregations, or domain knowledge. Consider combinations such as (Normalized A * One-Hot Encoded C) * B. Drop redundant features if they harm model performance. For each proposed feature, provide:

- Feature Name
- Justification: Why it improves {Objective}
- Samples

A.2 Role-play mutation and crossover

Your role is {Role}. Enhance predictions for {Objective} using a dataset: {Method of collection}.
You have recently tried these feature combinations with the following validation accuracy:

- "Norm. A", "One-Hot C" : {Accuracy}
- "B", "One-Hot Encoded C" : {Accuracy}
- "Norm. A", "B * One-Hot C" : {Accuracy}

Propose new feature combinations to improve prediction performance using transformations, interactions, aggregations, or domain knowledge. Consider combinations such as (Normalized A * One-Hot Encoded C) * B. Drop redundant features if they harm model performance. For each proposed feature, provide:

- Feature Name
- Justification: Why it improves {Objective}
- Samples: [...]

A.3 LTM evaluation

Enhance predictions for {Objective} using a dataset: {Method of collection}.
The dataframe df is loaded. Each column corresponds to an attribute:

- {Column A Name}: (Numerical, range: {Min}–{Max}), Samples: [...], {Description}
- {Column B Name}: (Boolean, {0 = X}, {1 = Y}), Samples: [...], {Description}
- {Column C Name}: (Categorical, {1 = X}, {2 = Y}, {3 = Z}), Samples: [...], {Description}

A data scientist has proposed creating a df with new features useful for TabPFN predicting Objective:

- Feature name / Justification / Samples
- Feature name / Justification / Samples

Write code to generate these additional columns in df, adhering to the feature descriptions and considering column types and class semantics. The classifier will train on the updated dataframe df.

A.4 Hyperparameter settings

50 iterations of random search optimized XGBoost, CatBoost, and LightGBM, while AutoGluon, H_2O, PyCaret, and TabPFN were used as off-the-shelf AutoML solutions.

Table 2 Hyperparameter settings

XGBoost	eta: {0.01, 0.05, 0.1}, n_estimators: {100, 200, 500}, max_depth: {3, 5, 7}, subsample: {0.5, 0.8, 1}, alpha: {0, 0.1, 1}, lambda: {1, 1.5, 2}, gamma: {0, 1, 5}, colsample_bytree: {0.5, 0.7, 1}, colsample_bylevel: {0.5, 0.7, 1}, min_child_weight: {1, 3, 5}
LightGBM	num_leaves: {31, 63, 127}, max_depth: {-1, 5, 10}, learning_rate: {0.01, 0.05, 0.1}, min_child_weight: {1, 3, 5}, reg_alpha: {0, 0.1, 1}, reg_lambda: {1, 1.5, 2}, n_estimators: {100, 200, 500}, subsample: {0.5, 0.8, 1}
CatBoost	learning_rate: {0.01, 0.05, 0.1}, iterations: {1000, 1500, 2000}, depth: {6, 10, 12}, l2_leaf_reg: {1, 3, 5}, border_count: {32, 64, 128}, subsample: {0.7, 0.8, 1}, random_strength: {0.5, 1, 2}, bagging_temperature: {0.5, 1, 2}