

LLM-Driven Performance-Space Augmentation for Meta-learning-Based Algorithm Selection*

Darren Zhu
e1121541@u.nus.edu

National University of Singapore
Singapore

Daren Ler
dcsdlsw@nus.edu.sg

National University of Singapore
Singapore

Abstract

Meta-learning for algorithm selection suffers from a persistent limitation: the number of curated real-world datasets is small, resulting in sparse meta-datasets that constrain meta-learner generalisation. In this paper, we address this problem by augmenting the meta-dataset with synthetic regression datasets produced via a large language model (LLM), with generation steered toward target regions of a low-dimensional *performance space*. Our experiments, which utilise 42 real-world UCI regression datasets and 730 synthetic datasets, show that a uniform augmentation policy significantly outperforms a no-augmentation policy, with a 17.47% relative reduction in Hamming loss, a 100.41% relative improvement in subset accuracy, and a 6.09% relative gain in pooled out-of-fold R^2 . These results lead us to postulate a central thesis. *The performance of algorithms resides on a low-dimensional performance manifold, whose reconstruction bias may be minimised by user-guided LLMs that seek to maximise uniform ϵ -cover, and consequently, lead to improved meta-learning for algorithm selection.*

CCS Concepts

• Computing methodologies → Machine learning.

Keywords

algorithm selection, performance prediction, meta-dataset data augmentation, large language models

ACM Reference Format:

Darren Zhu and Daren Ler. 2026. LLM-Driven Performance-Space Augmentation for Meta-learning-Based Algorithm Selection. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '26)*. ACM, New York, NY, USA, 8 pages.

1 Introduction

Algorithm selection corresponds to the problem of choosing the most suitable algorithm for a given dataset. As no single algorithm is universally optimal over all possible datasets [43, 42], we are required to match an algorithm’s inductive bias to the properties of the dataset. Meta-learning for algorithm selection provides a principled framework for this problem [29, 32].

As with most learning problems, the effectiveness of meta-learning depends critically on the data, or in this case, the meta-dataset. In practice, curated dataset repositories such as UCI [16] are used to populate such meta-datasets. However, they typically comprise a limited number of datasets, which limits the

generalisability of any model trained over such data [33], especially given that the meta-feature space is high-dimensional [27].

In this paper, we propose a different approach to **synthetic meta-dataset augmentation**: (1) we frame augmentation in *performance space*, a low-dimensional space defined by the cross-validated R^2 scores of candidate algorithms; (2) we compare two augmentation strategies, uniform- versus margin-based sampling, and (3) we utilise an LLM-based generator to produce synthetic regression datasets whose algorithmic behaviour falls within specified target regions of this performance space.

Our experiments, which utilise 42 real-world UCI regression datasets and 730 synthetically generated datasets, show that data augmentation substantially improves meta-learner generalisation over the unaugmented baseline, and that uniform augmentation consistently outperforms margin-based augmentation.

These findings support the viability of performance-space-guided augmentation and suggest that broad coverage is more valuable than concentrated boundary sampling for meta-learning.

2 Related Literature

2.1 Meta-Learning for Algorithm Selection

The algorithm selection problem may be decomposed into four components: (i) a problem space $\mathcal{P}$, (ii) a feature space $\mathcal{F}$, (iii) an algorithm space $\mathcal{A}$, and (iv) a performance space $\mathcal{Y}$ [29, 32]. A (base-level) dataset $x \in \mathcal{P}$ is described by a meta-feature vector $f(x) \in \mathcal{F}$, and the goal is to learn a selection mapping $S : \mathcal{F} \rightarrow \mathcal{A}$ such that the chosen algorithm $\alpha \in \mathcal{A}$ is expected to maximise performance $y(\alpha(x)) \in \mathcal{Y}$.

Useful taxonomies of meta-learning [17, 38] characterise prior work along three dimensions: meta-features, meta-learners, and meta-targets. Meta-features are defined as descriptors used to characterise datasets; meta-learners are the models used to map those descriptors to recommendations; and meta-targets specify the specific predictive objective at the meta-level. Meta-features are categorised into information-theoretic, statistical, and simple [8, 22], complexity-based [15], model-based [4], landmarking-based [5, 26, 23], and structural-information-based descriptors [37, 36]. Meta-learners are categorised into rule-based [2, 9], regression-based [28, 13, 6], instance-based [39, 46, 10], multi-label-based [40, 41], and link-prediction-based methods [47]. Meta-targets include best-algorithm prediction [13, 28, 6], ranked-list recommendation [10, 36], and multiple-algorithm recommendation [41, 40].

Despite the general framework’s applicability across domains, meta-learning for regression is less mature [24]. For regression, meta-features typically include classification-based meta-features, generic statistical measures such as variance, skewness, and kurtosis, algorithm-specific measures such as kernel target alignment

* Code is available at https://github.com/lxt3rm/llm_synth_generator.

[14, 34, 11], and landmarks [21, 35, 3]. More recently, Lorena et al. [24] proposed a suite of regression-specific complexity meta-features, demonstrating that such measures describe regression datasets more effectively than classical meta-features.

However, Soares [33] and Reif et al. [27] argue that the larger concern for meta-learning is the sparsity of real-world datasets relative to the high dimensionality of the meta-feature space.

2.2 Augmenting the Meta-Dataset

Soares [33] addresses the meta-dataset scarcity problem by deriving “datasetoids” from existing real-world datasets. These new datasets are generated by swapping categorical features with the class. Using 64 UCI datasets, 983 datasetoids were generated, which improved the meta-level accuracy for determining decision-tree pruning.

Alternatively, Reif et al. [27] proposes the use of a genetic data generator, where candidate datasets are evolved so that their measured meta-features match user-specified target values. This in turn allows synthetic datasets to be placed more deliberately in under-represented parts of the meta-feature space to improve coverage.

More recently, Komorniczak [19] proposed the use of a genetic algorithm, called Evolutionary Projection-based Complexity Optimisation (EPCO), that transforms synthetically generated source datasets so that selected complexity measures approach desired target values. Developed for both classification and regression, EPCO was able to produce datasets with varying target difficulty, which correlate with downstream recognition quality.

Collectively, these studies suggest a progression toward more structured approaches to augmenting the meta-dataset, i.e., addressing *which specific augmentation policy would produce the most informative meta-examples for algorithm selection*. This gap defines the positioning of our work.

2.3 LLM for Synthetic Tabular Generation

Recent research has demonstrated that Large Language Models (LLMs) can be used effectively for synthetic tabular data generation [44]. Existing methods can be broadly grouped into prompt-based approaches [18, 12, 31], fine-tuned tabular generators [7], and hybrid structured systems [45]. Prompt-based methods rely on instruction design, in-context exemplars, and conditional prompting to improve realism and long-tail coverage, whereas fine-tuned approaches, such as GReaT, train autoregressive models directly on serialised samples to improve structural consistency. Hybrid methods further combine LLM reasoning with external components for constraint handling and statistical alignment.

Collectively, these studies show that LLMs are promising generators of tabular data, especially when augmented with mechanisms that enforce validity and dependency constraints. However, these methods target instance-level augmentation, whereas the focus for meta-learning is the dataset level, i.e., we seek to generate entire synthetic datasets to induce controlled variation in downstream algorithm performance.

3 Proposed Synthetic Dataset Generation

The proposed method frames synthetic dataset generation in the landmark performance space, i.e., in a subset of $\mathcal{Y}$, rather than the typical meta-feature space $\mathcal{F}$. As each axis of $\mathcal{Y}$ encodes a candidate

algorithm’s performance on a dataset, augmentations that target a subset of $\mathcal{Y}$ would correspond, in part, to algorithm-selection decisions. More specifically, we propose that augmentation targets the $[0, 1]^2$ space defined by the cross-validated R^2 scores of two anchor algorithms, KNN and LR.

3.1 Hypotheses

A meta-learner’s predictive quality is bounded by both its learning algorithm and the structural coverage of its training meta-dataset. When that meta-dataset is only built on available real-world regression datasets, problem regimes are under-represented and out-of-sample generalisation degrades. Synthetic augmentation therefore matters for filling under-covered regions of the problem space, not for inflating meta-dataset size. We frame this coverage in $\mathcal{Y}$ following Ler et al. [23], who showed that algorithm “domains of expertise” inhabit a lower-dimensional projection than meta-features suggest.

H1: Synthetic-augmentation hypothesis. Augmenting the *real-world meta-dataset* (the meta-dataset comprising only curated real-world datasets) with synthetic regression datasets improves the meta-learner’s out-of-sample generalisation. The null hypothesis is that augmentation does not improve out-of-sample performance over the real-world meta-dataset alone; the alternate is that it does.

H2: Sampling-strategy hypothesis. The choice of sampling strategy within performance-space-targeted augmentation affects meta-learner performance. The two-dimensional performance space is spanned by the cross-validated R^2 scores of the two anchor algorithms, and the *decision boundary* is the diagonal $y = x$ where both algorithms perform equally. The null hypothesis is that uniform and margin-based sampling produce equivalent out-of-sample meta-learner performance; the alternate is that they differ.

3.2 Proposed LLM Synthetic Dataset Generator

An unguided generator that samples raw (X, y) pairs achieves the targeted coverage only in the limit, and at intractable cost. We define an ϵ -cover as the condition that every point of the targeted region has a synthetic dataset *witness* within distance ϵ . We discretise $\mathcal{Y}$ into a rectangular grid and cast augmentation as populating selected cells. The mapping $\phi(D) = (R_{\text{KNN}}^2, R_{\text{LR}}^2) \in [0, 1]^2$ produces *structural embedding* coordinates under a fixed 5-fold cross-validation harness. A dataset is added when $\phi(D)$ lies inside the target cell.

The generator achieves this through a *propose–execute–evaluate–repair* loop, illustrated in Figure 1. We pose generation as a constrained program-synthesis problem, so the LLM (GPT 5.4) returns an executable Python procedure $\text{generate}(\text{seed}) \rightarrow (X, y)$ rather than raw data. A prompt specifies the target intervals for the current cell. Then, each proposed generator runs in a sandboxed subprocess that exposes an $\text{evaluate}(X, y)$ hook. The hook lets the procedure run a lightweight within-procedure search over its hyperparameters and return the candidate whose measured ϕ -coordinates lie closest to the target. If the candidate’s ϕ -coordinates fall outside the target cell, the system issues a *repair prompt* reporting the achieved coordinates back to the LLM. This is the dashed feedback arrow in Figure 1, sent as a continuation of the same conversational

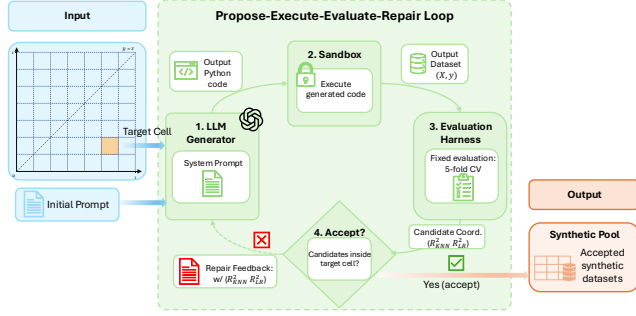


Figure 1: The propose-execute-evaluate-repair loop.

thread. Retries reuse the same execution seed to isolate architectural changes from stochastic variation. The system collects $w = 10$ witnesses per cell under a total budget of $b = 84$ attempts.

3.3 Meta-Dataset Generation

The meta-dataset has one row per regression dataset, with a 12-dimensional meta-feature vector and two parallel meta-label formulations (continuous and binary).

Meta-feature generation. We compute the 12 complexity measures of Lorena et al. [24] via problemxity [20] after standard preprocessing (one-hot encoding categoricals, removing missing values, min-max normalising). The measures cover Feature Efficiency (C1–C4), Linearity (L1–L3), Local Smoothness (S1–S4), and Dimensionality (T2). This yields a 12-dimensional vector $f(x) \in \mathcal{F}$.

Regression meta-label generation. The candidate set comprises five scikit-learn [25] regressors: K-Nearest Neighbours, Linear Regression, Lasso, Ridge, and Elastic Net. Per-dataset hyperparameter optimisation uses Optuna [1] with a TPE sampler and median pruner. The search spaces are: (1) KNN, $n_neighbors \in \{1, \dots, \lfloor n/3 \rfloor\}$, $weights \in \{uniform, distance\}$; (2) Ridge and Lasso, $\alpha \sim \log \mathcal{U}(10^{-5}, 10^1)$; (3) Elastic Net, $\alpha \sim \log \mathcal{U}(10^{-5}, 10^1)$, $l1_ratio \sim \mathcal{U}(10^{-3}, 1 - 10^{-3})$; (4) linear regression at default.

The tuning objective is the mean R^2 over a 10-fold inner *Totally Stratified Cross-Validation* (TSCV) [30], which round-robin-assigns sorted targets to folds so that target distributions match across folds. The selected configuration is then fixed and re-evaluated under a separate 10-fold outer TSCV repeated 10 times. The continuous meta-label is the mean of the resulting outer-split scores,

$$y_{\text{reg}}^{(a)}(D) = \frac{1}{|S_{\text{out}}|} \sum_{s \in S_{\text{out}}} R_s^2(a).$$

Because tuning runs once per dataset and is reused across outer splits, the protocol is a *two-stage tune-then-evaluate* procedure rather than fold-wise nested optimisation.

Multi-label meta-label generation. The multi-label formulation reuses the outer-split performance matrix. Algorithms are ranked per split by descending R^2 , with average ranks for ties. A Friedman chi-square test at $\alpha = 0.05$ checks for differences across the algorithm pool. If its null is not rejected, all five candidates are

applicable. Otherwise, we compute the Nemenyi *Critical Distance*

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}},$$

with $k = 5$, $N = 100$, and q_α the studentised range statistic. An algorithm receives $y_{\text{ml}}^{(a)}(D) = 1$ when its average rank satisfies $r_a \leq r^* + CD$, where r^* is the best (lowest) average rank, and 0 otherwise. The vector then identifies the algorithms statistically indistinguishable from the top-ranked method.

3.4 Synthetic Augmentation Strategy

Accepted synthetic datasets are mapped into the same meta-feature and meta-label representation as the real-world datasets (see Section 3.3). Augmentation is applied fold-wise. For each training split, n synthetic rows are sampled and appended to the training fold, while the test fold remains exclusively real-world. This procedure operationalises H1, and the two strategies below operationalise H2.

- **Uniform Strategy (Global-Cover Sampler).** The n synthetic rows are sampled without replacement, with equal probability irrespective of position in $\mathcal{Y}$.
- **Margin-Based Strategy (Boundary-Concentrated Sampler).** For synthetic dataset j at coordinates $(s_{x,j}, s_{y,j})$, the perpendicular distance to the diagonal $y = x$ is

$$d_j = \frac{|s_{x,j} - s_{y,j}|}{\sqrt{2}}.$$

Sampling probabilities are a temperature-scaled softmax over $-d_j$, so smaller d_j yields larger probability. The n datasets are sampled without replacement.

4 Experimental Setup

We evaluate two meta-learning formulations on the meta-feature space $\mathcal{F}$ defined in Section 3.3. The real-world meta-dataset comprises 42 regression datasets from the UCI repository, each described by the 12 complexity measures of Lorena et al. [24]. The synthetic augmentation pool is encoded in the same space.

As per Section 3.2, synthetic datasets are generated in a $\mathcal{Y} = (R_{\text{KNN}}^2, R_{\text{LR}}^2)$ performance space over a 5x5 grid, yielding 248 datasets, and a 7x7 grid, yielding 482, giving us a synthetic pool of 730.

The regression formulation predicts each candidate algorithm's held-out R^2 following Lorena et al. [24]. We train one Transformed TargetRegressor per algorithm, each pipeline standardising features and using a default SVR base. Targets are standardised within each fold, and no hyperparameter tuning is performed.

The multilabel formulation [40, 41] predicts a binary meta-label vector via binary relevance, with MultiOutputClassifier(GaussianNB()) preceded by feature standardisation. One independent classifier is trained per label, sans calibration or threshold tuning.

Both formulations share a meta-learner evaluation protocol, distinct from the meta-label TSCV in Section 3.3. We run shuffled $K = 10$ cross-validation with 10 repeats over a 10-seed sweep. For a fixed seed, folds are shared across the no-augmentation baseline, the Uniform Strategy, and the Margin-Based Strategy, isolating the

augmentation effect. Augmentation is applied fold-wise to the training split, and synthetic datasets never participate in split formation or held-out evaluation.

The Uniform Strategy and Margin-Based Strategy are defined in Section 3.4. The Margin-Based softmax temperature is $\alpha = 10$, with sampling probabilities computed once per evaluation run and reused across folds and repeats. Both strategies sample without replacement.

4.1 Evaluation Metrics

For the regression formulation following Lorena et al. [24], we report a pooled out-of-fold R^2 per algorithm.

$$R_{\text{OOF},k}^2 = 1 - \frac{\sum_i (y_{ik} - \hat{y}_{ik})^2}{\sum_i (y_{ik} - \bar{y}_k)^2},$$

where y_{ik} is the held-out performance of algorithm k on dataset i , $\hat{y}_{ik}$ the within-repeat across-fold pooled prediction, and $\bar{y}_k$ the mean target across real-world datasets. The pooling distinguishes this from a fold-averaged R^2 . Additionally, for learning-curve analysis, we average across the five targets to obtain `mean_target_pooled_oof_r2`.

For the multilabel formulation, we pool hard out-of-fold predictions within a repeat and report subset accuracy and Hamming loss. Subset accuracy is the fraction of datasets where all $K = 5$ binary labels match. Hamming loss is the mean per-label error and is our primary multilabel metric, since subset accuracy is dominated by its exact-match stringency.

Within a seed, repeat-level scores are summarised by mean and standard deviation. Across seeds, we report the seed-mean, seed-standard-deviation, and mean within-seed standard deviation. Learning-curve plots show mean $\pm$ one seed-standard-deviation bands.

4.2 Comparison Protocol

An improvement is a higher pooled OOF R^2 for the regression formulation, or lower Hamming loss for the multilabel formulation. We report fixed-budget summaries (at $n_{\text{syn}} = 400$) and learning-curve summaries varying n_{syn} . The learning-curve sweep tests H1 at its endpoints and H2 across the full n_{syn} range.

Significance is assessed by a two-sided paired Student's t -test (`scipy.stats.ttest_rel`) on per-seed scores at $\alpha = 0.05$. The shared-fold property makes pairing valid. For H1, the test runs once per metric at the learning-curve endpoint ($n_{\text{syn}} = 730$), where the two strategies collapse to one arm, yielding three p -values. For H2, the test runs at every train-size point and we report the count of significant points (out of 37) per metric. All p -values are uncorrected.

5 Analysis of Results

5.1 Ablation Analysis and Learning Curves

The Uniform and Margin-Based Strategies populate the performance space differently, and that difference shapes the downstream comparisons that follow. We define a canonical instance set per strategy as the top $n_{\text{syn}} = 400$ candidates by selection frequency across 1000 Monte Carlo draws of 400 from the synthetic pool of

730. Figure 2 projects the two canonical sets into $\mathcal{Y}$. The Margin-Based set concentrates in a tight band along the decision boundary $y = x$, while the Uniform set spans the unit square.

Ablation study. Across all three metrics the Uniform Strategy attains the best performance at the fixed budget $n_{\text{syn}} = 400$, and both strategies improve over the no-augmentation baseline (Table 1). The protocol holds evaluation and augmentation volume constant, so the comparison isolates strategy choice.

Per-formulation results. For the multilabel formulation, the Uniform Strategy reduces Hamming loss from 0.4050 to 0.3342 (17.47% relative) while the Margin-Based Strategy reaches 0.3531. Subset accuracy rises from 0.1752 to 0.3512 under uniform, a 100.41% gain over baseline and 35.20% over margin; the lower absolute value reflects exact-match stringency over $K = 5$ binary labels rather than poor model quality. For the regression formulation, pooled OOF R^2 rises from 0.7646 to 0.8112 under uniform (6.09%) and to 0.7940 under margin (3.84%). The SVR-based regression meta-learner is less sensitive to strategy than the Naive Bayes classifier. The H1 endpoint paired t -test at $n_{\text{syn}} = 730$ confirms augmentation against the baseline at $p < 10^{-14}$ for all three metrics.

Strategy comparison. The Uniform Strategy's advantage stems from three sources. First, the Margin-Based softmax temperature $\alpha = 10$ leaves the strategy non-tightly concentrated, with instances reaching $d = 0.2$ from $y = x$. Second, the meta-learners benefit more from distributional diversity than from concentrated boundary signal at this budget. Third, margin sampling collapses joint variation in $(R_{\text{KNN}}^2, R_{\text{LR}}^2)$ across the augmented training rows. Drawing only rows where the two scores agree forces near-collinearity, which reduces the independent information the two coordinates carry. The Uniform Strategy by contrast preserves the full range. The synthesis pool itself is bounded; the 482 accepted generators draw from a narrow numerical-operator vocabulary dominated by `np.sin` (50%), `np.exp` (22%), and `np.cos` (12%).

Learning curve setup. We sweep n_{syn} from 20 to 730 in steps of 20 under the no-augmentation baseline, the Uniform Strategy, and the Margin-Based Strategy, yielding 37 train-size points per metric. Figure 3 shows the resulting Hamming loss, subset accuracy, and pooled OOF R^2 curves with ± 1 standard-deviation bands across seeds.

Per-panel results. At $n_{\text{syn}} = 20$ both strategies underperform the baseline on Hamming loss in a cold-start regime (Uniform 0.425, Margin-Based 0.409, baseline 0.405; Figure 3(a)). Uniform crosses the baseline near $n_{\text{syn}} \approx 80$ and plateaus near 0.330 between 460 and 500; margin plateaus later. Both converge to 0.327 at 730. The subset accuracy panel (Figure 3(b)), by contrast, is non-monotonic, with Uniform peaking at 0.352 near 380 and declining to 0.326, and Margin-Based peaking at 0.338 near 560. The divergence from monotonic Hamming implies that at high volumes the classifier gains per-label accuracy but loses joint-label calibration. The pooled OOF R^2 panel (Figure 3(c)) has no cold-start. Both strategies improve over the baseline (0.765) at $n_{\text{syn}} = 20$ and rise monotonically to 0.821 at 730. Error bands contract from $\sigma \approx 0.003$ to 0.0006. Across all three panels, the H2 paired t -test favours uniform at $p < 0.05$ for 28 of 37 points on Hamming loss, 30 of 37 on subset accuracy,

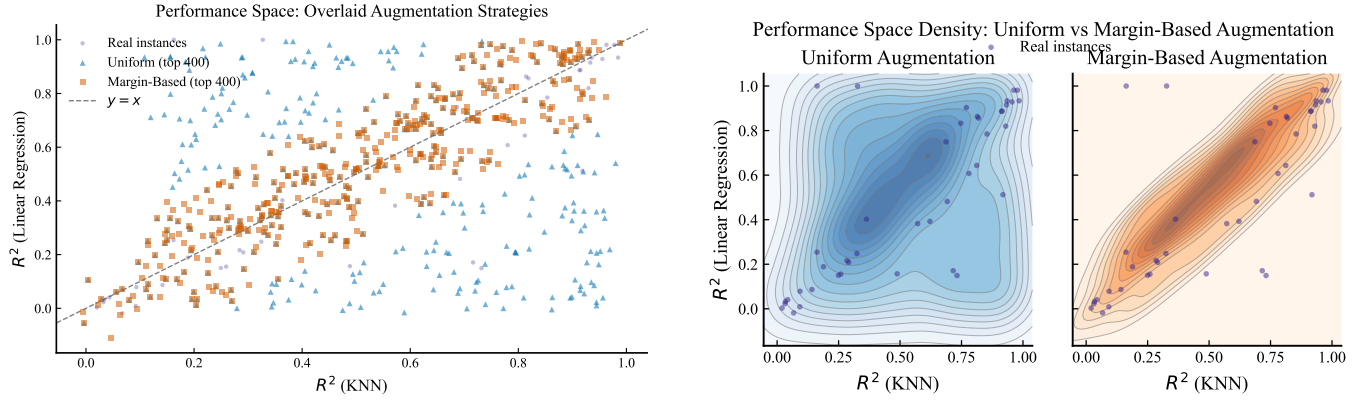


Figure 2: Uniform (blue) and margin-based (orange) canonical instance sets in $(R^2_{\text{KNN}}, R^2_{\text{LR}})$ performance space. (a) The margin-based set clusters tightly along the $y = x$ diagonal, while the uniform set spans the full space. (b) The margin-based density is concentrated in a narrow ridge, whereas the uniform density extends into off-diagonal regions.

Table 1: Ablation study on data augmentation strategies across tasks. *Uniform* samples uniformly at random from the augmentation pool; *Margin-based* selects augmentations near the decision boundary. Relative gains in the last two columns are computed as $\Delta = (v_{\text{new}} - v_{\text{base}})/|v_{\text{base}}|$, with signs reflecting the optimisation direction.

Task	Metric	Direction	No Aug.	Margin	Uniform	Unif. vs Margin	Unif. vs No Aug.
Multilabel Classification							
	Hamming Loss	↓	0.4050	0.3531	0.3342	+5.34%	+17.47%
	Subset Acc.	↑	0.1752	0.2598	0.3512	+35.20%	+100.41%
Regression							
	Pooled OOF R^2	↑	0.7646	0.7940	0.8112	+2.17%	+6.09%

and 34 of 37 on R^2 . Reversals are confined to $n_{\text{syn}} \in [560, 580]$ on the multilabel metrics and to the convergence end on R^2 .

Cross-curve summary. The three panels jointly support H1’s alternate and H2’s alternate; augmentation improves over the no-augmentation baseline, and Uniform dominates Margin-Based across most of the sweep. The convergence at $n_{\text{syn}} = 730$ to identical scores, where both strategies necessarily draw the entire pool, serves as a consistency check on the experimental pipeline. Table 1’s fixed-budget point ($n_{\text{syn}} = 400$) sits near the subset-accuracy peak, before the Hamming plateau, and mid-ascent on R^2 . Within the budget regime tested, distributional diversity in the augmentation set drives generalisation more than concentrated boundary signal.

5.2 Implications: Geometry of Generalisation

The consistent advantage of Uniform over Margin-Based sampling suggests that meta-learning is a process of reconstructing the functional topography of the performance space rather than identifying decision boundaries. We interpret this advantage through a structural framework centred on the relationship between the synthetic data and the geometric structure of real-world dataset performance.

The Performance Manifold and Universal Addressing. The structural embedding ϕ defined in Section 3.2 assigns every dataset D a universal address based on its performance characteristics. This address sits in a landmarker performance space (LPS) whose axes carry the performance of simple landmarker algorithms, in this study KNN and LR. The LPS is a 2-D projection that we treat as the performance manifold reconstructed during meta-learning. Although the final meta-learning target is 5-D, our results suggest that the underlying structural logic of the datasets is captured within the 2-D LPS.

Topological Covering and Resolution Matching. The findings imply that meta-learner success depends on topological covering, i.e., synthetic points must spread across the manifold so as to leave no significant gaps. We formalise this through an ε -cover condition. Let $\mathcal{X}$ denote the space of real-world regression datasets and $\mathcal{S}$ the synthetic pool. The synthetic meta-dataset $\mathcal{S}$ provides an ε -cover of $\mathcal{X}$ when

$$\forall x \in \mathcal{X}, \exists s \in \mathcal{S} : \|\phi(x) - \phi(s)\|_2 < \varepsilon,$$

so the synthetic training data brackets the performance characteristics of any real-world dataset. Uniform sampling outperforms

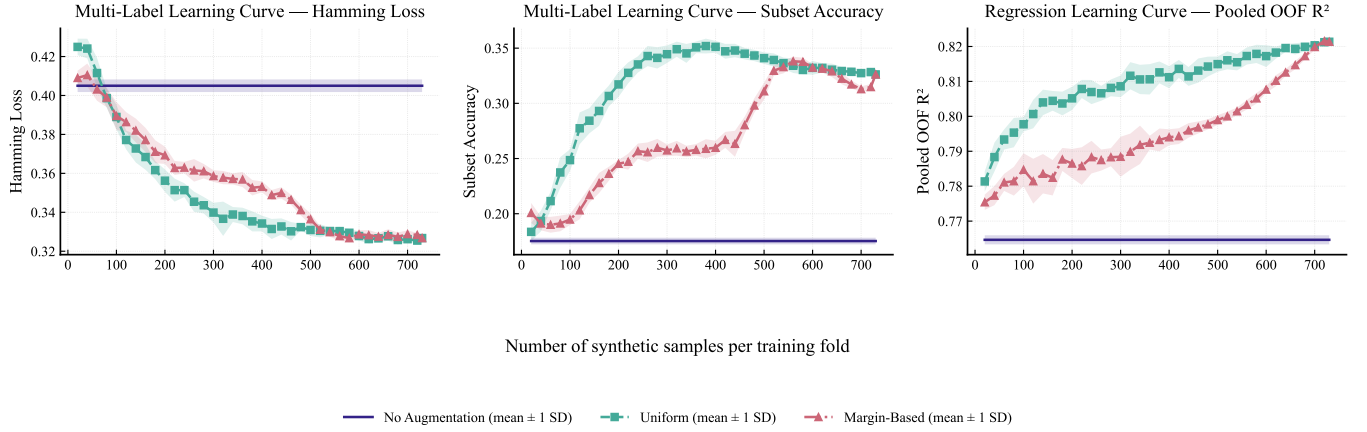


Figure 3: Meta-learner learning curves as a function of the number of synthetic samples per training fold n_{syn} , under the no-augmentation baseline, the Uniform Strategy, and the Margin-Based Strategy. The results depict the (a) Hamming loss (lower is better), (b) Subset accuracy (higher is better), and (c) Pooled OOF R^2 (higher is better). Each curve plots the mean across 10 seeds; shaded bands show ± 1 standard deviation.

margin-based sampling because it minimises manifold reconstruction bias; where margin sampling focuses on ambiguous areas, uniform ensures a global cover that prevents blind spots, enabling resolution matching between synthetic-data density and the structural resolution of the manifold.

Steerable Synthesis and the Propose–Execute–Evaluate–Repair Loop. The capacity to achieve this cover within a finite computational budget is driven by steerable synthesis, an LLM’s ability to generate datasets with specific, targeted properties. We operationalise this through the generation loop from Section 3.2.

The loop bridges the LLM and the manifold, populating the coordinates required for high-fidelity meta-learning with synthetic witness datasets. The relative failure of margin-based refinement confirms that the meta-learner needs a complete functional topography, not just a refined view of the boundaries. The narrow significant reversals in the multilabel panels around $n_{\text{syn}} \approx 600$ suggest that the optimal strategy may be volume-dependent, shifting from broad coverage to boundary refinement once the manifold is sufficiently populated.

6 Conclusion

In this work, we began with an investigation into mitigating the scarcity of meta-learning data through LLM-driven synthesis. However, our findings have evolved into a broader thesis regarding the nature of algorithm selection. More specifically, we notice that while individual dataset complexity matters, **the global coverage of the performance manifold is the primary driver of meta-learner generalisation.**

We demonstrated that while LLMs generate structurally simpler datasets, these datasets provide a critical “scaffold” for the meta-learner by acting as a representative sample of the functional landscape. Our comparison of sampling strategies revealed a clear hierarchy: **Uniform Performance-Space Sampling** consistently outperforms margin-based approaches. This suggests that for algorithm selection, the meta-learner requires a *holistic functional*

topography. Achieving an empirical ε -cover of the performance manifold is the primary prerequisite for generalisation, as it allows the meta-learner to anchor its structural scaffolding before attempting local boundary refinement. By prioritising this “covering” over high-density refinement near decision boundaries, we achieved a 100.41% improvement in subset accuracy for multi-label selection.

6.1 Future Work

Firstly, to overcome the identified structural limits where the generator favours additive mixtures, we propose a *Prompt Chaining* architecture. Instead of monolithic dataset generation, the process would be decomposed into a three-stage sequence.

- (1) **Logic Selection:** Including non-linear operators (e.g., `np.where` for branching or `np.linalg` for spectral transforms).
- (2) **Implementation:** Script drafting based on selected logic.
- (3) **Validation:** Executing the code to ensure the resulting performance profile aligns with the target cell in the ε -cover.

This “*Chain-of-Synthesis*” architecture ensures a fuller traversal of structural complexity within the algorithm selection space.

A second natural extension is a wider sweep of grid granularity beyond the two grids used here. With the uniform-augmentation benefit already established, finer grids would test whether meta-learner generalisation improves monotonically with cell density, saturates at some resolution, or reverses. A finer sweep would also identify where, if anywhere, a single granularity dominates jointly across the regression and multilabel formulations.

However, our immediate focus in the near future will be to work on a **theoretical formalisation of ε -covers**. Our most immediate next steps will be the development of a formal theoretical framework to quantify the density of the synthetic meta-dataset as an ε -cover of the performance manifold. This would allow us to determine the minimum number of synthetic datasets required to guarantee a specific bound on meta-learning performance over unseen real-world datasets.

References

- [1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: a next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631.
- [2] Shawkat Ali and Kate A Smith. 2006. On learning algorithm selection for classification. *Applied Soft Computing*, 6, 2, 119–138.
- [3] M Fatih Amasyali and Okan K Ersoy. 2009. A study of meta learning for regression. *ECE Technical Reports*, 386.
- [4] Hilan Bensusan. 1998. God doesn't always shave with occam's razor—learning when and how to prune. In *European Conference on Machine Learning*. Springer, 119–124.
- [5] Hilan Bensusan and Christophe Giraud-Carrier. 2000. Discovering task neighbourhoods through landmark learning performances. In *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 325–330.
- [6] Hilan Bensusan and Alexandros Kalousis. 2001. Estimating the predictive accuracy of a classifier. In *European Conference on Machine Learning*. Springer, 25–36.
- [7] Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. 2022. Language models are realistic tabular data generators. *arXiv preprint arXiv:2210.06280*.
- [8] Pavel Brazdil, João Gama, and Bob Henery. 1994. Characterizing the applicability of classification algorithms using meta-level learning. In *European conference on machine learning*. Springer, 83–102.
- [9] Pavel Brazdil, João Gama, and Bob Henery. 1994. Characterizing the applicability of classification algorithms using meta-level learning. In *European conference on machine learning*. Springer, 83–102.
- [10] Pavel B Brazdil, Carlos Soares, and Joaquim Pinto Da Costa. 2003. Ranking learning algorithms: using ibl and meta-learning on accuracy and time results. *Machine Learning*, 50, 3, 251–277.
- [11] Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz Kandola. 2001. On kernel-target alignment. *Advances in neural information processing systems*, 14.
- [12] Liancheng Fang, Aiwei Liu, Hengrui Zhang, Henry Peng Zou, Weizhi Zhang, and Philip S Yu. 2025. Tabgen-icl: residual-aware in-context example selection for tabular data generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, 20027–20041.
- [13] Luis PF Garcia, Ana C Lorena, Stan Matwin, and André CPLF de Carvalho. 2016. Ensembles of label noise filters: a ranking approach. *Data Mining and Knowledge Discovery*, 30, 5, 1192–1216.
- [14] Taciana AF Gomes, Ricardo BC Prudêncio, Carlos Soares, André LD Rossi, and André Carvalho. 2012. Combining meta-learning and search techniques to select parameters for support vector machines. *Neurocomputing*, 75, 1, 3–13.
- [15] Tin Kam Ho and Mitra Basu. 2002. Complexity measures of supervised classification problems. *IEEE transactions on pattern analysis and machine intelligence*, 24, 3, 289–300.
- [16] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. [n. d.] The uci machine learning repository. Retrieved Apr. 17, 2026 from <https://archive.ics.uci.edu>.
- [17] Irfan Khan, Xianchao Zhang, Mobashar Rehman, and Rahman Ali. 2020. A literature survey and empirical study of meta-learning for classifier selection. *IEEE Access*, 8, 10262–10281.
- [18] Jinhee Kim, Taesung Kim, and Jaegul Choo. 2024. Epic: effective prompting for imbalanced-class data synthesis in tabular data classification via large language models. *Advances in Neural Information Processing Systems*, 37, 31504–31542.
- [19] Joanna Komorniczak. 2025. Transforming datasets to requested complexity with projection-based many-objective genetic algorithm. *arXiv preprint arXiv:2507.15132*.
- [20] Joanna Komorniczak and Paweł Kseniewicz. 2023. Proplexity—an open-source python library for supervised learning problem complexity assessment. *Neurocomputing*, 521, 126–136.
- [21] Petr Kuba, Pavel Brazdil, Carlos Soares, and Adam Woznica. 2002. Exploiting sampling and meta-learning for parameter setting for support vector machines.
- [22] Jun Won Lee and Christophe Giraud-Carrier. 2008. Predicting algorithm accuracy with a small set of effective meta-features. In *2008 Seventh International Conference on Machine Learning and Applications*. IEEE, 808–812.
- [23] Daren Ler, Irena Koprinska, and Sanjay Chawla. 2004. A landmarker selection algorithm based on correlation and efficiency criteria. In *Australasian Joint Conference on Artificial Intelligence*. Springer, 296–306.
- [24] Ana C Lorena, Aron I Maciel, Péricles BC De Miranda, Ivan G Costa, and Ricardo BC Prudêncio. 2018. Data complexity meta-features for regression problems. *Machine Learning*, 107, 1, 209–246.
- [25] F. Pedregosa et al. 2011. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- [26] Bernhard Pfahringer, Hilan Bensusan, and Christophe Giraud-Carrier. 2000. Meta-learning by landmarking various learning algorithms. In *ICML*, 743–750.
- [27] Matthias Reif, Faisal Shafait, and Andreas Dengel. 2012. Dataset generation for meta-learning. *KI-2012: Poster and Demo Track*, 69–73.
- [28] Matthias Reif, Faisal Shafait, Markus Goldstein, Thomas Breuel, and Andreas Dengel. 2014. Automatic classifier selection for non-experts. *Pattern Analysis and Applications*, 17, 1, 83–96.
- [29] John R Rice. 1976. The algorithm selection problem. In *Advances in computers*. Vol. 15. Elsevier, 65–118.
- [30] José A Sáez and José L Romero-Béjar. 2022. Impact of regressand stratification in dataset shift caused by cross-validation. *Mathematics*, 10, 14, 2538.
- [31] Nabeel Seedat, Nicolas Huynh, Boris Van Breugel, and Mihaela Van Der Schaar. 2023. Curated llm: synergy of llms and data curation for tabular augmentation in low-data regimes. *arXiv preprint arXiv:2312.12112*.
- [32] Kate A Smith-Miles. 2009. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, 41, 1, 1–25.
- [33] Carlos Soares. 2009. Uci++: improved support for algorithm selection using datasetoids. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 499–506.
- [34] Carlos Soares and Pavel B Brazdil. 2006. Selecting parameters of svm using meta-learning and kernel matrix-based meta-features. In *Proceedings of the 2006 ACM symposium on Applied computing*, 564–568.
- [35] Carlos Soares, Pavel B Brazdil, and Petr Kuba. 2004. A meta-learning method to select the kernel width in support vector regression. *Machine learning*, 54, 3, 195–209.
- [36] Qinqiao Song, Guangtao Wang, and Chao Wang. 2012. Automatic recommendation of classification algorithms based on data set characteristics. *Pattern recognition*, 45, 7, 2672–2689.
- [37] Nikolaj Tatti. 2007. Distances between data sets based on summary statistics. *Journal of Machine Learning Research*, 8, Jan, 131–154.
- [38] Joaquin Vanschoren. 2019. Meta-learning. In *Automated machine learning: methods, systems, challenges*, 35–61.
- [39] Guangtao Wang, Qinqiao Song, Heli Sun, Xueying Zhang, Baowen Xu, and Yuming Zhou. 2013. A feature subset selection algorithm automatic recommendation method. *Journal of Artificial Intelligence Research*, 47, 1–34.
- [40] Guangtao Wang, Qinqiao Song, Xueying Zhang, and Kaiyuan Zhang. 2014. A generic multilabel learning-based classification algorithm recommendation method. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9, 1, 1–30.
- [41] Guangtao Wang, Qinqiao Song, and Xiaoyan Zhu. 2015. An improved data characterization method and its application in classification algorithm recommendation. *Applied Intelligence*, 43, 4, 892–912.
- [42] David Hilton Wolpert. 1996. The existence of a priori distinctions between learning algorithms. *Neural Computation*, 8, 7, 1391–1420.
- [43] David Hilton Wolpert. 1996. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8, 7, 1341–1390.
- [44] Kaituo Zhang et al. 2026. The llm data auditor: a metric-oriented survey on quality and trustworthiness in evaluating synthetic data. *arXiv preprint arXiv:2601.17717*.
- [45] Mingming Zhang, Zhiqing Xiao, Guoshan Lu, Sai Wu, Weiqiang Wang, Xing Fu, Can Yi, and Junbo Zhao. 2025. Aigt: ai generative table based on prompt. In *Proceedings of the 31st International Conference on Computational Linguistics*, 9926–9938.
- [46] Xueying Zhang, Ruixian Li, Bo Zhang, Yunxiang Yang, Jing Guo, and Xiang Ji. 2019. An instance-based learning recommendation algorithm of imbalance handling methods. *Applied Mathematics and Computation*, 351, 204–218.
- [47] Xiaoyan Zhu, Xiaomei Yang, Chenchen Ying, and Guangtao Wang. 2018. A new classification algorithm recommendation method based on link prediction. *Knowledge-Based Systems*, 159, 171–185.

Reproducibility

All code, prompts, and cached artefacts are publicly available in two repositories; their READMEs document the full command sequences, pinned dependency versions, configuration knobs, and runtime estimates, so we summarise only the structure here. The generator repository, at https://github.com/lxt3rm/llm_synth_generator, implements the propose–execute–evaluate–repair loop of Section 3.2 and regenerates the synthetic pool end-to-end from the fixed prompts and a deterministic per-cell seed schedule—the 5×5 grid of 248 datasets, the 7×7 grid of 482, and the combined pool of 730 (this stage requires an OpenAI API key). The evaluation repository, at https://github.com/lxt3rm/meta_learner_experiments, consumes that pool together with the real-world meta-dataset of

42 UCI regression datasets and runs the no-augmentation baseline, the Uniform Strategy, and the Margin-Based Strategy under the shared-fold protocol of the Experimental Setup, reproducing the per-seed scores, paired t -test tables, and learning-curve figures of the Analysis of Results.

Two properties keep reproduction lightweight. First, the evaluation repository ships the cached synthetic meta-datasets (248, 482, and 730 rows) together with a copy of the 42 UCI regression datasets, so every downstream result can be reproduced without re-running the LLM or fetching data externally. Second, both repositories are pure Python and CPU-only, with pinned versions of scikit-learn, numpy, scipy, and pandas (plus matplotlib for the figures). The original datasets are available from the UCI Machine Learning Repository’s public listings.