SimBA: Simplifying Benchmark Analysis Using Performance Matrices Alone

Nishant Subramani[♠*] **Alfredo Gomez**[♠*] **Mona Diab**[♠]

[♠]Carnegie Mellon University - Language Technologies Institute
{nishant2,alfredo3,mdiab}@cs.cmu.edu

Abstract

Modern language models are evaluated on large benchmarks, which are difficult to make sense of, especially for model selection. Looking at the raw evaluation numbers themselves using a model-centric lens, we propose **SimBA**, a three phase framework to **Sim**plify **B**enchmark **A**nalysis. The three phases of **SimBA** are: `stalk`, where we conduct dataset & model comparisons, `prowl`, where we discover a representative subset, and `pounce`, where we use the representative subset to predict performance on a held-out set of models. Applying **SimBA** to three popular LM benchmarks: HELM, MMLU, and BigBenchLite reveals that across all three benchmarks, datasets and models relate strongly to one another (`stalk`). We develop an representative set discovery algorithm which *covers* a benchmark using raw evaluation scores alone. Using our algorithm, we find that with 6.25% (1/16), 1.7% (1/58), and 28.4% (21/74) of the datasets for HELM, MMLU, and BigBenchLite respectively, we achieve coverage levels of at least 95% (`prowl`). Additionally, using just these representative subsets, we can both preserve model ranks and predict performance on a held-out set of models with *near zero* mean-squared error (`pounce`). Taken together, **SimBA** can help model developers improve efficiency during model training and dataset creators validate whether their newly created dataset differs from existing datasets in a benchmark. Our code is open source, available at https://github.com/nishantsubramani/simba.

1 Introduction

The rapid expansion of language model (LM) benchmarks has resulted in an overabundance of evaluation datasets. However, the relationships among these datasets remain poorly understood. Current evaluation methods primarily focus on overall model win rates or simple aggregate measures, which fail to provide fine-grained insights into dataset characteristics and model performance trends (Liang et al., 2023). One approach that the community has taken to mitigate this problem is to look at instance-level predictions and construct coresets, where each individual dataset in a benchmark is subsampled using various heuristics (Rodriguez et al., 2021; Perlitz et al., 2024; Zouhar et al., 2025). This resulting subset is used as a proxy for the entire benchmark. Coreset identification, often done using influence functions (Koh and Liang, 2017; Schioppa et al., 2021), has many drawbacks: integration into an already existing evaluation framework is hard, weak statistical signal hinders generalization, and collecting instance-level predictions across many models may be computationally infeasible (Chatterjee and Hadi, 1986).

Our work looks to uncover a more structured and simplified understanding of benchmarks through an analysis of the datasets and models directly from the performance matrix *without* collecting any instance-level predictions. Our framework to **Sim**plify **B**enchmark **A**nalysis is called **SimBA** and has three phases:

1. **`Stalk`**: Analyzing relationships between datasets and measuring how models relate to one another across a benchmark.
2. **`Prowl`**: Discovering a representative subset of datasets from a benchmark that maintains model order.
3. **`Pounce`**: Predicting model performances using the representative set based on performance patterns.

Using our three phase approach in Figure 1, we analyze HELM (Liang et al., 2023), MMLU (Hendrycks et al., 2020), and BigBenchLite (Srivastava et al., 2022) and find that both datasets and models correlate well with one another (§2). Motivated by this, we find that:

1. We can identify representative subsets S_{HELM},

[*]Equal Contribution

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 13220–13233
November 4-9, 2025

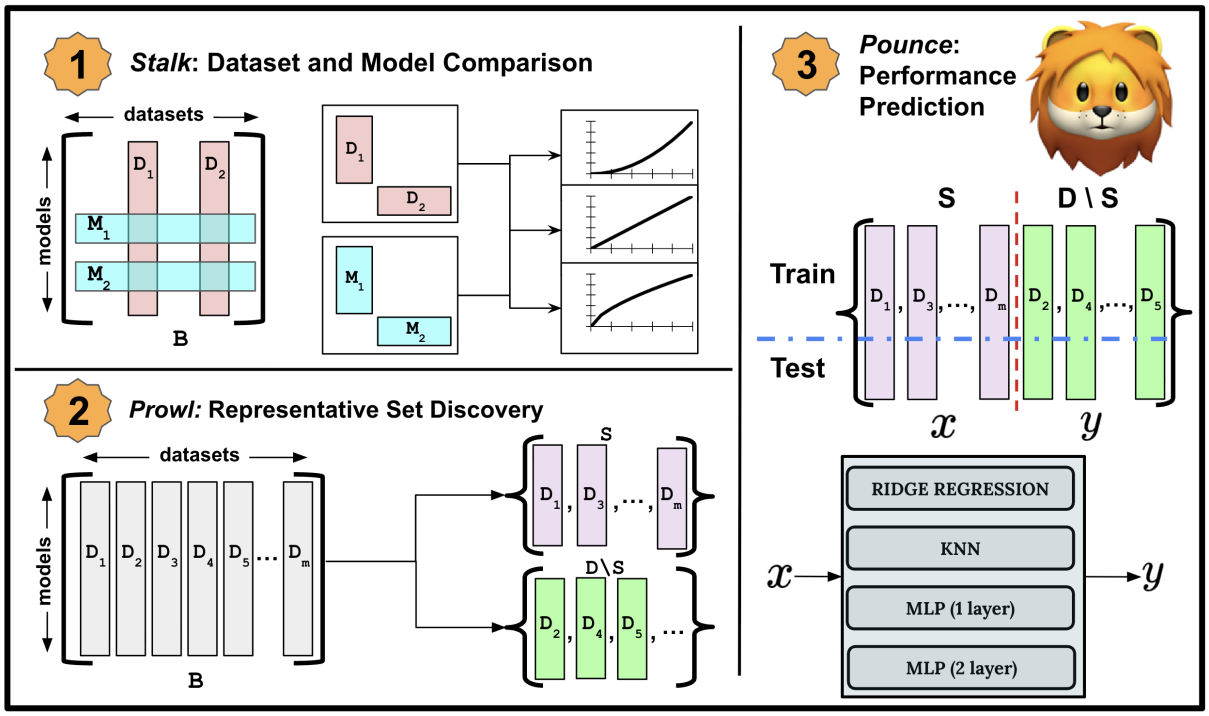

Figure 1: An overview of SimBA, our three phase analysis framework. Its three phases are: `stalk`: dataset & model comparison, `prowl`: representative dataset discovery, and `pounce`: performance prediction.

S_{MMLU}, and S_{BBL} with just 6.25% (1/16), 1.7% (1/58), and 28.4% (21/74) of datasets respectively that achieve greater than 95% coverage (§3).

2. Our representative subsets preserve model ranks and can predict performance on a held-out set of models with *near zero* error (§4).

Taken together, our three phase analysis, **SimBA**, can be used directly by language model practitioners and dataset developers alike to improve efficiency and efficacy.

2 `Stalk`: Dataset & Model Comparison

A benchmark is represented as a matrix $B \in \mathbb{R}^{m \times d}$, where m is the number of models and d is the number of datasets. B can have missing values. Different datasets evaluate different metrics, which often are scaled differently (*e.g.* classification vs. generation), so we normalize B. For every dataset D_i with random chance performance x_{random}, we modify every observation $x_1, \ldots, x_m$ to be:

$$x_j = max\left(0, \frac{x_j - x_{random}}{1 - x_{random}}\right) \quad (1)$$

Note these new x_j values correspond to percent above random chance. We use this normalization for all analysis. [1] Moreover, all values are further normalized to be within the interval $[0, 1]$, where 0 corresponds to random chance and 1 corresponds to the maximum possible performance.

2.1 Dataset Comparison

Datasets are the essence of a benchmark and the first step in our recipe is to compare datasets. For every pair of datasets D_i and D_j, we compare their performance numbers to identify how they relate to each other using one of four relationships: LINEAR, EXPONENTIAL, POWER-LAW, or NONE. To measure this, we learn a multi-variate linear regressor f with parameters $W_{reg} \in \mathbb{R}^{m \times 1}$ and $b \in \mathbb{R}$ to optimize the objective:

$$B[:, j] = W_{reg}^T \cdot B[:, i] + b \quad (2)$$

Here, $M[:, i]$ and $M[:, j]$ are the performance numbers across all models for datasets D_i and D_j. We then obtain the R^2 value to quantify the amount of variation explained by the regressor. This naturally works to establish the strength of a linear relationship. To measure an EXPONENTIAL or POWER-LAW relationship, we apply a transformation to the measurements of one dataset before learning the lin-

[1] Srivastava et al. (2022) use the same normalization for BigBench for some of their analyses.

ear regressor. [2] Since we are predicting one dataset from another, we learn a total of *6* regressors: two per type of relationship (LINEAR, EXPONENTIAL, POWER-LAW), one were D_j is predicted from D_i and vice-versa. We choose the one with the largest R^2 value. If that $R^2 < 0.5$ between two datasets, we classify its relationship as NONE. [3]

2.2 Model Comparison

We compare models across a benchmark in the same way as datasets: for every pair of models M_1, M_2, we follow the procedure in §2.1 to classify whether the relationship is one of four types: LINEAR, EXPONENTIAL, POWER-LAW, or NONE. Crucially, we only use performance numbers across the benchmark to make this classification. [4] Identical to dataset comparison, we learn *6* regressors and choose the relationship with the largest R^2 value. We classify the relationship between two models to be NONE if the best regressor results in an $R^2 < 0.5$, just as in §2.1.

3 `Prowl`: Representative Dataset Discovery

Equipped with an understanding of a benchmark, our next step is to identify a target. More specifically, we want to discover a representative subset of a benchmark S from the performance matrix $B \in \mathbb{R}^{m \times d}$ alone, where m is the number of models and d is the total number of datasets.

3.1 Dataset Similarity

To discover a representative subset, we need a method to determine whether a dataset is redundant. [5] We compare datasets based on model performance patterns alone using 9 similarity measures: three correlations (PEARSON, SPEARMAN, KENDALL-TAU) and six similarities (COSINE, MANHATTAN, EUCLIDEAN, MINKOWSKI (P=3), WASSERSTEIN, and JENSEN-SHANNON). [6] We compute similarities between all pairs of datasets, resulting in a matrix $C_{\text{SIM}} \in \mathbb{R}^{d \times d}$, where each entry is a similarity score based on a similarity measure SIM.

3.2 Discovering a Representative Subset

First, we define a proxy metric for the coverage of a candidate representative set S. The PROXY_COVERAGE (δ) of S under a similarity measure SIM is computed as:

$$\delta(S, \text{SIM}) = \frac{\sum_{i \in D} \lambda_i}{|\mathcal{D}|} \tag{3}$$

where λ_i is defined as:

$$\lambda_i = \begin{cases} 1, & \text{if } i \in S \\ \max_{j \in S} C_{\text{SIM}}[i, j], & \text{otherwise} \end{cases} \tag{4}$$

Additionally, we need to define the COVERAGE_GAIN (Ψ) of a set S when a dataset D_i is added under a similarity measure SIM. This is used as a heuristic to measure how much PROXY_COVERAGE (δ) is gained when adding a new dataset D_i to S:

$$\Psi(S, D_i) = \delta(\{S, D_i\}, \text{SIM}) - \delta(S, \text{SIM}) \tag{5}$$

Equipped with these measures, we propose Algorithm 1 to discover a representative subset S from a collection of datasets $\{D_1, \ldots D_d\}$. While our implementation supports beam search for representative dataset discovery, empirical evaluation showed that larger beam widths (5, 10, 20) provided no meaningful improvement over the greedy approach across HELM, MMLU, and BigBenchLite. [7]

Algorithm 1 Representative Dataset Discovery

input $C_{\text{SIM}} \in \mathbb{R}^{d \times d}$, γ, $\mathcal{D} = \{D_1, \ldots, D_d\}$
output S

```
S ← ∅
while δ(S) < γ ∧ |S| ≤ d do
    D* ← arg max_{D_i ∈ D\S} Ψ(S, D_i)
    S ← S ∪ {D*}
end while
return S
```

[2] This step resembles applying a nonlinear kernel (*e.g.*, radial-basis function kernel) to an SVM to learn a nonlinear relationship.

[3] $R^2 = 0.5$ indicates that only 50% of the variation is explained by the regressor, which we deem is too low to confidently claim a specific relationship. This threshold is a tunable hyperparameter.

[4] We explicitly do not use any information about the model architecture, size, or family, but one could to improve upon the model comparison phase.

[5] If there exists a set of datasets $\{D_1, \ldots, D_k\}$ that are redundant, we need to keep only one in our representative set.

[6] All similarities are computed as: SIM = 1 - DISTANCE or SIM = $\exp^{\text{DISTANCE}}$ based on whether DISTANCE is bounded. See Appendix A for more details.

[7] γ is a PROXY_COVERAGE threshold set by the user. A value of 1 would entail iteratively building S until it contains all the datasets in the benchmark $\{D_1, \ldots D_d\}$.

Baselines: We also evaluate the following baselines as random and simple baselines in dataset selection have been shown to be strong (Diddee and Ippolito, 2025). RANDOM is a baseline where S is populated with a random dataset iteratively without replacement. We average across 1000 random runs in our experiments. GREEDY MINIMUM is a baseline where S is populated with the dataset with the lowest average performance across all models. GREEDY MAXIMUM is a baseline where S is populated with the dataset with the highest average performance across all models. For both greedy baselines, S is also populated iteratively.

3.3 Representative Subset Evaluation

Using Algorithm 1, we discover a representative subset S that has n datasets. To measure how well S covers the original benchmark B, we first find the mean win rate (MWR) of each model as compared to the other models based entirely on S:

$$\text{MWR}(B') = \frac{\sum_{\mu \leq m, d \leq n} \mathbb{I}[B'[\mu, d] > B'[\mu', d]]}{m - 1} \tag{6}$$

Remember that $B' \in \mathbb{R}^{m \times n}$, where m is the number of models and n is the number of datasets in S. B' is matrix formed by collecting all performance numbers for all models for the datasets in S, so $\text{MWR}(B') \in \mathbb{R}^m$. We then compute the PEARSON-CORRELATION between $\text{MWR}(B')$ and the mean-win-rate obtained on the full benchmark, $\text{MWR}(B) \in \mathbb{R}^m$, to obtain coverage η:

$$\eta(B') = \text{PEARSON}(\text{MWR}(B), \text{MWR}(B')) \tag{7}$$

Identifying S is a greedy process involving a similarity function SIM. Since S iteratively increases from having one to two to many datasets, we require an algorithm to identify how good a similarity function SIM is in discovering a strong representative subset at every size. Additionally, we also require a method to compare two similarity functions SIM_1 and SIM_2. Taking inspiration from the receiver-operating characteristic curve and taking the area under it (AUROC) (Marcum, 1960), we develop our own signed AUC measure called *subset coverage AUC* (SCAUC). [8] To compute SCAUC, we iteratively build S one dataset at a time until we get to the full benchmark ($S = \{D_1, \ldots, D_d\}$). Starting with the first dataset chosen ($|S| = 1$), we compute $\eta(B')$ using equation 7. [9] B' is the matrix formed by taking all the performance numbers for all the datasets in S from the original benchmark matrix B. We then construct a curve using the d coverage values (η) and compute the signed area under that curve. [10] To compare two similarity functions, we measure their SCAUC values and choose the one with a higher value.

4 Pounce: Performance Prediction

Equipped with the results of the first two phases of our analysis pipeline, we predict the performances directly using a representative subset. Since the representative subset S is just a subset of the full benchmark B, we evaluate whether different regression based approaches can predict $D \setminus S$ (*i.e.*, the subset of D that is not in S) from S alone.

One general approach for matrix prediction is Singular Value Decomposition (SVD; Candes and Recht (2009)). However, SVD works only on a partially observed matrix, where rows (models) and columns (datasets) have at least one observation. In a realistic setting, we want to use our subset S to predict the other dataset performances $\mathcal{D} \setminus S$ on entirely new models, so SVD would not be immediately applicable. As a result, we focus on three regressor types: RIDGE REGRESSION, KNN REGRESSION, and MLP REGRESSION.

Regularized Linear Regression RIDGE REGRESSION uses a linear function with L2 regularization penalty between a model's performance scores on S and its performance on $D \setminus S$. This regularization helps prevent overfitting when training on small representative subsets (Hoerl and Kennard, 1970; Hastie et al., 2009).

KNN Regression KNN REGRESSION estimates the performance score y for a dataset by averaging the performance scores of its k nearest neighbors in feature space as follows:

$$y = \frac{1}{k} \sum_{i \in \mathcal{N}_k(X)} y_i \tag{8}$$

Here $\mathcal{N}_k(X)$ denotes the set of the k closest datasets to X using a chosen distance metric (*e.g.*, EUCLIDEAN) (Altman, 1992). We use $k = 5$ neighbors, or the size of the training set if smaller than 5. Additionally, KNN REGRESSION does not impose

[8] Subramani et al. (2025) also develop a signed AUC measure to measure the tool-calling utility of LLMs.

[9] Note: $\eta(\emptyset)$ is undefined and $\eta(B) = 1$.

[10] This is signed area because correlations can be negative.

a functional form, allowing it to capture non-linear relationships.

MLP Regression A Multi-Layer Perceptron (MLP) is a feedforward neural network that models non-linear relationships using multiple layers (Rosenblatt, 1958). We experiment with two MLP architectures a single hidden layer with 12 neurons and a two-layer architecture with 12 neurons in each hidden layer. We include MLP REGRESSION because it can capture complex non-linear relationships between dataset features and model performance.

4.1 Performance Prediction Evaluation

To evaluate how well we can predict performance, mean squared error (MSE) is a natural choice. For a given representative set S, we train a regressor to predict performance on the remaining datasets in the benchmark ($\mathcal{D} \setminus S$). [11] We compute the MSE of the regressor on the held-out test set on ($\mathcal{D} \setminus S$).

In our experiments, we build S sequentially, by greedily adding one dataset at a time according to Algorithm 1. As a result, we can measure MSE at each point for $|S| = 1, ..., |S| = |\mathcal{D}| - 1$. [12] This traces an MSE curve. Using a similar approach to tracing the area under the coverage curve like in §3, we can compute the area under the MSE curve, which we term `AUC-MSE`. [13] Note that high values of `AUC-MSE` indicate high error because this curve is an error curve *not* a performance curve like other AUC curves. To measure which regressor is best, we compare `AUC-MSE` values and choose the method with the lowest `AUC-MSE` value.

5 Experiments

Benchmarks For our analysis, we look at three benchmarks: HELM (Liang et al., 2023), MMLU (Hendrycks et al., 2020), and BigBench-Lite (Srivastava et al., 2022). We look at the core scenarios of HELM (17 datasets, 29 models), MMLU (58 datasets, 79 models), and BigBench-Lite (74 datasets, 45 models), splitting each benchmark into training and test sets to validate our analysis. All datasets are included in both splits, but models are separated across training and test sets randomly with 80% of models in training and 20% of the models in test. This means that HELM has 23 models in train and 6 models in test, MMLU has 63 models in train and 16 in test, and BigBench-Lite has 36 models in train and 9 in test. For each benchmark, we go through the 3 analysis phases from Figure 1, discuss those results in §6, and expand the analysis to look at robustness in §7.

Relationships	HELM	MMLU	BBLite
LINEAR	6	67	754
EXPONENTIAL	25	649	139
POWER-LAW	44	810	143
NONE	45	127	1665
Total	120	1653	2701

Table 1: Dataset comparison results with counts between each pair of datasets and their classifications as LINEAR, EXPONENTIAL, POWER-LAW, and NONE based on the highest R^2 value. See §2.1 for details.

6 Results

6.1 `Stalk`: Dataset & Model Comparison

Dataset Comparison: We measure how datasets relate to one another using the methodology in §2.1. Table 1 shows that for HELM, only 5% of the pairs of datasets have a LINEAR relationship, while 37.5% lack any relationship, indicating minor redundancy among datasets.

For MMLU, 4.1% of dataset pairs have a LINEAR relationship and only 7.7% lack a relationship, suggesting that most datasets are predictive of one another through non-linear relationships (EXPONENTIAL: 39.3%, POWER-LAW: 49.0%). For BigBenchLite (BBLite), 27.9% of dataset pairs have a LINEAR relationship while 61.6% show no relationship at all, indicating that BBLite has the most diverse and independent datasets among the three benchmarks. Taken together, we suspect that finding a small representative dataset would be most difficult for BBLite.

Model Comparison: We measure how models relate to one another via the method in §2.2. Table 2 shows that for HELM, only 4.9% of model pairs have LINEAR relationships, with the majority showing POWER-LAW relationships (72.4%). MMLU demonstrates more complex model relationships with 9.4% LINEAR, 40.8% EXPONENTIAL, and 42.3% POWER-LAW relationships. BigBenchLite shows 7.6% LINEAR relationships and

[11] $\mathcal{D}$ is the set of datasets in the benchmark $\mathcal{D} = \{D_1, ..., D_d\}$.

[12] When $|S| > |\mathcal{D}| - 1$, ($|\mathcal{D} \setminus S| = 0$).

[13] `AUC-MSE` is not signed because the minimum error one can get is 0.0, so every point on this curve is in the top right quadrant of a cartesian coordinate plane ($x, y > 0$).

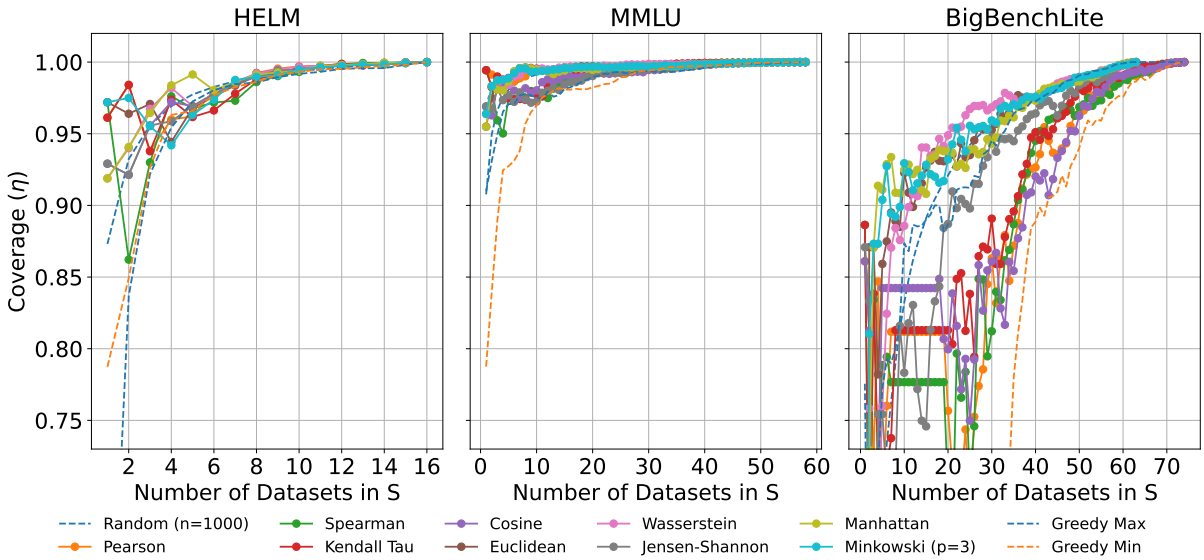

Figure 2: Here, we measure the coverage (η) across HELM, MMLU, and BigBenchLite as our representative subset S grows. We report performance for all three baselines and all nine similarity measures discussed in §3.

Relationships	HELM	MMLU	BBLite
LINEAR	20	290	75
EXPONENTIAL	72	1258	41
POWER-LAW	294	1302	549
NONE	20	231	325
Total	406	3081	990

Table 2: Model comparison results with counts between each pair of datasets and their classifications as LINEAR, EXPONENTIAL, POWER-LAW, and NONE based on the highest R^2 value. See §2.2 for details.

55.5% POWER-LAW relationships. Notably, all three benchmarks have relatively few model pairs with no discernible relationship (HELM: 4.9%, MMLU: 7.5%, BigBenchLite: 32.8%). BigBenchLite again has the highest rate of no relationships further hinting that discovering a representative subset for BigBenchLite may be the most difficult.

6.2 `Prowl`: Coverage Analysis

Our goal is to determine the minimum number of datasets needed to achieve a specific coverage level η on a specific benchmark. Since we experiment with nine similarity functions, we want to identify the best similarity measure for this task. To do this, we first compare every pair of datasets D_i, D_j using each of the similarity functions defined in §3. This results in a similarity matrix $C_{\text{SIM}} \in \mathbb{R}^{d \times d}$ for each similarity function. [14] On each similarity matrix C_{SIM}, we apply our coverage algorithm (Algorithm 1) using its respective similarity function and construct S. We measure coverage (η) using equation 7 at each iteration as S is being constructed until $S = \mathcal{D}$. This traces a coverage curve and we measure the area under this coverage curve and report this SCAUC value in Table 3. We also report the size of the *smallest* representative subset $S*$ that achieves a coverage $\eta \geq 0.95$.

We find that we can achieve coverage levels of at least 95% with just 6.25% (1/16), 1.7% (1/58), and 28.4% (21/74) of the datasets for HELM, MMLU, and BigBenchLite respectively. This represents a substantial efficiency gain: particularly for MMLU and HELM, where a single well-chosen dataset can effectively represent nearly the entire benchmark for model ranking purposes. Additionally, Table 3 shows that the choice of similarity measure has varying effects across benchmarks. For HELM and MMLU, most similarity measures perform similarly well, with several achieving the optimal single-dataset representative subset. However, for BigBenchLite, there is more variation in performance, with WASSERSTEIN achieving the best results (|S*| = 21) and several measures like PEARSON and SPEARMAN requiring substantially more datasets (|S*| = 42 and 41 respectively).

Finally, using Algorithm 1 with most similarity measures outperforms all baselines on average

[14] C_{SIM} could be an upper (or lower) triangular matrix because our similarity functions are symmetric.

Similarity Methods	HELM SCAUC (↑)	HELM $\lvert S^*\rvert$(↓)	MMLU SCAUC (↑)	MMLU $\lvert S^*\rvert$(↓)	BigBenchLite SCAUC (↑)	BigBenchLite $\lvert S^*\rvert$(↓)
RANDOM (n = 1000)	0.980	2.5	0.994	2.3	0.928	25.2
GREEDY MINIMUM	0.967	4	0.980	8	0.607	51
GREEDY MAXIMUM	0.957	4	0.988	4	0.933	33
PEARSON	0.984	**1**	**0.997**	**1**	0.886	42
SPEARMAN	0.975	**1**	0.992	**1**	0.884	41
KENDALL-TAU	0.983	**1**	0.994	**1**	0.899	40
COSINE	0.981	3	0.992	**1**	0.896	48
MANHATTAN	**0.985**	3	0.996	**1**	0.945	32
EUCLIDEAN	0.984	**1**	0.996	**1**	0.943	27
MINKOWSKI (P=3)	0.983	**1**	0.996	**1**	**0.950**	22
WASSERSTEIN	0.983	3	**0.997**	**1**	0.943	**21**
JENSEN-SHANNON	0.980	3	0.991	**1**	0.903	36

Table 3: Performance of our three baselines and seven similarity measures on the identification of a representative subset task for HELM, MMLU, and BigBenchLite. SCAUC is the area under the coverage curves present in Figure 2. S^* is the smallest subset that achieves $\eta(S^*) = 0.95$. **Bold** indicates the best performing system for each metric.

Regressor	AUC−MSE (↓) HELM	MMLU	BBLite
RIDGE	0.005	**0.002**	**0.002**
KNN	**0.004**	**0.002**	**0.002**
MLP (1 layer)	0.081	0.110	0.006
MLP (2 layer)	0.031	0.081	0.006

Table 4: Performance as measured by AUC−MSE across HELM, MMLU, and BigBenchLite. We use MINKOWSKI (P=3) as the similarity measure SIM to identify representative subsets S for our four regressors: RIDGE, KNN, MLP (1 layer), and MLP (2 layer). AUC−MSE is the area under the curves in the top row of Figure 3. **Bold** indicates the best performing method for each benchmark.

across all three benchmarks, though the improvement is less pronounced for BigBenchLite due to its more diverse dataset composition. We also compute how often using Algorithm 1 with a similarity measure outperforms the RANDOM baseline. See Table 6 for details on the proportion of times a system outperforms the random baseline across 1000 random runs.

6.3 `Pounce`: Performance Prediction

Our goal in this phase is to validate that representative datasets enable accurate performance prediction. Having identified efficient representative subsets in phase II (`prowl`), we now assess whether performance on these subsets can predict performance on the remaining datasets. We first split our models into training (80%) and test (20%) sets. Using only the models in the training set, we identify representative datasets at 80% coverage and train four performance predictors: RIDGE REGRESSION, KNN REGRESSION, and MLP REGRESSION with one and two layers. We then evaluate these predictors on the test set of models, measuring their ability to predict scores (MSE) for the remaining datasets based solely on performance patterns observed in the representative subset.

As shown in Table 4, both RIDGE REGRESSION and KNN REGRESSION perform exceptionally well across all three benchmarks, achieving *near zero* AUC−MSE values. RIDGE REGRESSION achieves (0.005, 0.002, 0.002) and KNN REGRESSION achieves (0.004, 0.002, 0.002) for HELM, MMLU, and BigBenchLite respectively. Meanwhile, Figure 3 shows that prediction error generally decreases as the representative subset size increases, but with diminishing returns. The MLP models consistently underperform, with the single-layer MLP showing particularly poor results on HELM and MMLU. Additionally, in Figure 3, we find that KNN REGRESSION achieves negligible error with just *one* dataset on both HELM and MMLU. This could suggest that these benchmarks are saturated.

For MMLU, all four regressors maintain consistently low error rates across different subset sizes, with RIDGE and KNN maintaining the lowest error

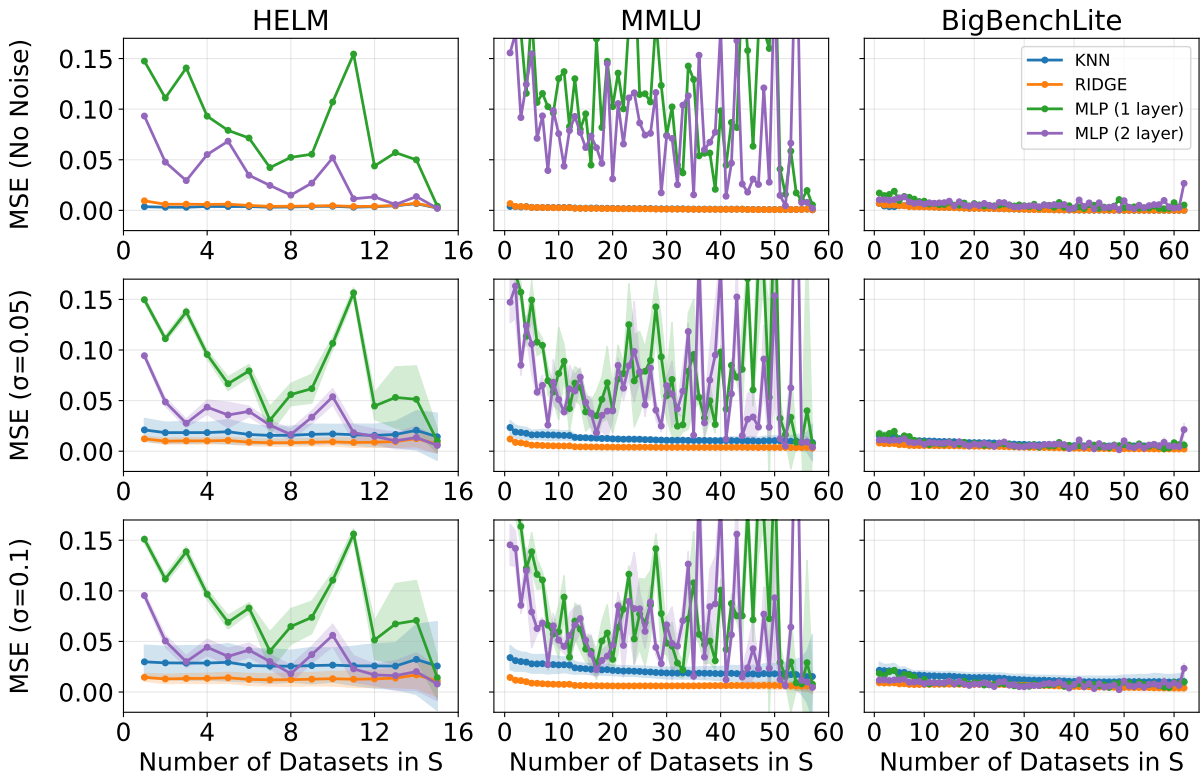

Figure 3: We measure the mean squared error (MSE) on a held-out test set of models of regressors trained on a representative subset S using MINKOWSKI (P=3) as SIM. This is measured for HELM, MMLU, and BigBenchLite for the three regressors we experiment with in §4. Additionally, we repeat this experiment after adding two magnitudes of noise to the training set of B according to §7.1 ($\sigma = 0.05$ and $\sigma = 0.1$). Lower scores are better.

rates throughout. This complements the analysis presented in §2.1, where MMLU showed the highest proportion of inter-dataset relationships, making it the most predictable benchmark.

7 Analysis

7.1 Is `pounce` robust to noise?

Our evaluation framework relies on point estimates of the performances of models on individual datasets. As such, robustness to noise is a critical consideration when evaluating different approaches of performance prediction. We evaluate the robustness of performance prediction by perturbing the training set of the benchmark matrix B_{train} as $B'_{train} = B_{train} + \mathcal{N}(\mu, \sigma^2)$ where μ is mean and σ is the noise level parameter. For all noise perturbations we use $\mu = 0$ and $\sigma^2 = 0.05$ or $\sigma^2 = 0.1$. This evaluation framework enables us to quantify the stability of our methods under varying noise conditions.

As shown in Figure 3, generally, error rates for all methods increase with greater noise. However, we observe that both RIDGE and KNN REGRESSION achieve low error rates under both noise conditions, across all three benchmarks (`AUC-MSE` < 0.03). RIDGE consistently maintains `AUC-MSE` values of 0.013 for all benchmarks and noise conditions. Predictably, the MLP systems are the least robust, but maintain similar error rates to the no noise version. In other words, the MLP systems just are not great at performance prediction.

7.2 Is `pounce` robust to varying data splits?

Here, we look at how different subsets or selections of data change the `pounce` stage. This is related to the noise level analysis in §7.1, but requires its own investigation. The former approximates uncertainty for individual observations, whereas here, we care about measuring the variance of `pounce` when the training and test splits vary.

To keep the test set uncontaminated, we look at just the training set. On this, we perform k-fold cross validation and measure performance via `AUC-MSE`. [15]. We observe low standard deviations across different training splits (< 0.01) indicating that performance prediction is stable. Intuitively,

[15] We use 5-fold cross validation for HELM and 10-fold for MMLU and BigBenchLite.

	Regressor	AUC−MSE (↓) HELM	MMLU	BBLite
$\sigma = 0.05$	RIDGE	**0.010**	**0.004**	**0.004**
	KNN	0.017	0.012	0.013
	MLP (1 layer)	0.081	0.079	0.007
	MLP (2 layer)	0.031	0.068	0.007
$\sigma = 0.1$	RIDGE	**0.013**	**0.007**	**0.006**
	KNN	0.027	0.021	0.013
	MLP (1 layer)	0.087	0.078	0.009
	MLP (2 layer)	0.033	0.067	0.008

Table 5: Performance as measured by AUC−MSE across HELM, MMLU, and BigBenchLite in the presence of noise (§7.1). We use MINKOWSKI (P=3) as the similarity measure SIM to identify representative subsets S for our four regressors: RIDGE, KNN, MLP (1 layer), and MLP (2 layer). AUC−MSE is the area under the curves in Figure 3. **Bold** indicates the best performing method for each benchmark.

the simpler models, RIDGE and KNN, are the most stable as they are the least prone to overfitting. Taken together, our analysis suggests that similar models tend to perform similarly on related tasks, hinting that local neighborhood relationships are effective for `pounce`. Overall our error rates remain small and are robust to perturbations or changes in both the training and evaluation sets.

8 Related Work

The NLP community increasingly evaluates on large benchmarks (Srivastava et al., 2022; Li et al., 2023; Liang et al., 2023). Some approaches attempt to make evaluation more efficient by doing instance-level reduction (Vivek et al., 2023; Polo et al., 2024; Perlitz et al., 2024). Magnusson et al. (2025) look at a loosely related problem and use small scale experiments for pretraining data selection. They, too, require instance-level information, but are focused on analyzing how training on different subsets of pretraining data affect performance rather than looking at a representative set of pretraining corpora. We differ in that we focus on aggregate metrics and do not need access to any instance-level information for efficiency and performance prediction.

As mentioned earlier, most work looks at identifying coresets of benchmarks at the instance-level and there exist numerous methods to do this across many different application areas (Lewis and Catlett, 1994; Killamsetty et al., 2021; Paul et al., 2021; Moser et al., 2025). Ye et al. (2023) is one of the few works that, like us, goes beyond instance-level work and tackles the performance prediction task using simple regressors on BigBench. However, their approach is not as lightweight as ours because they requires features of the model and datasets for accurate prediction.

From the field of psychometrics and measurement theory, there exists the idea of convergent and divergent validity (Campbell and Fiske, 1959). Convergent validity suggests that metrics measuring similar underlying constructs should correlate highly with each other. Conversely, discriminant validity indicates that metrics capturing fundamentally different aspects should show minimal correlation. Xiao et al. (2023) propose MetricEval, a framework motivated by measurement theory to conceptualize and evaluate the reliable and valid of natural language generation metrics.

9 Conclusion

We propose a three phase approach to **Sim**plify **B**enchmark **A**nalysis called **SimBA**: `stalk` (dataset & model comparison), `prowl` (representative set discovery), and `pounce` (performance prediction). Using our approach, we analyze the HELM, MMLU, and BigBenchLite benchmarks. Our analysis shows that models and datasets alike correlate well with one another (`stalk`). Additionally, using Algorithm 1, we can identify representative subsets with 1, 1, and 21 datasets respectively that achieve a greater than 95% coverage (`prowl`). These representative sets preserve the original model ranks on the benchmark and can be used to predict performance on held-out models with *negligible* error (`pounce`). Furthermore, **SimBA** can be used by LM practitioners and dataset developers directly to reduce evaluation costs and validate dataset uniqueness.

10 Limitations

Dataset & Model Comparison Our analysis assumes that the relationships between datasets are sufficiently stable over time. As models continue to improve and scale, the nature of these relationships may evolve, potentially requiring periodic reassessment of representative subsets. The approach provides a snapshot analysis based on current model performance matrices, but doesn't account for how these relationships might change with fundamentally new architectures or training paradigms.

Moreover, the presented analyses requires having a sufficient number of models evaluated on the benchmarks. If the available models lack in diversity in their underlying architectures, training data, or training methodologies, the identified relationships may not generalize to future models.

Identifying the Representative Dataset Our method is a greedy approach that iteratively chooses datasets such that PROXY_COVERAGE increases. A different, albeit more expensive, approach could exhaustively identify the best combination of datasets using a better search algorithm. We experimented with adapting to beam search, but found no significant improvement, perhaps because PROXY_COVERAGE is correlated, but can be slightly disconnected with COVERAGE depending on the similarity function used.

Performance Prediction Although KNN performs reasonably well, its AUC−MSE values are consistently 2-3 times higher than RIDGE in the presence of noise. In these settings the MLP models perform the worst with AUC−MSE values 5-10 times higher than RIDGE, possibly due to overfitting on the limited training data. Models trained with better regularization on more data would have greater stability and be less prone to overfitting, so this is something we recommend for practitioners using **SimBA**.

Overall Risks Our evaluation framework offers a three stage approach to better understand a benchmark. Although representative sets get high coverage, there could be cases when the representative set gets high coverage by chance. In this case, it would be risky to make major decisions that affect users based on a small sample of data.

11 Ethical Considerations

Since SimBA does not involve training generative models, the primary ethical concerns center on potential misuse of our framework's insights and the risk of overconfidence in representative subset evaluations.

Our finding that small representative subsets can achieve high coverage creates opportunities for manipulation. Model developers could strategically evaluate only on datasets where their models perform well, then use **SimBA** to claim coverage over the entire benchmark without actually testing on challenging datasets. This could mislead the research community and downstream users about true model capabilities. Similarly, the choice of similarity measure presents another avenue for selective reporting, as our analysis shows that different similarity functions (PEARSON, MINKOWSKI (P=3), WASSERSTEIN, etc.) can yield different representative subsets and coverage results.

To aid in mitigating these concerns, we recommend transparent reporting of representative set selection methodologies, evaluation across multiple correlation methods, and validation on diverse datasets. SimBA does not aim to replace comprehensive evaluation, especially for high-stakes deployments. Rather, it serves as a supplementary tool for understanding benchmark structure and improving evaluation efficiency in appropriate contexts.

Acknowledgments

We thank Iz Beltagy, Pradeep Dasigi, Dirk Groeneveld, and Alexis Ross for feedback on very early scoping of this work. Additionally, we thank Harshita Diddee, Athiya Deviyani, and members of MD's R3Lit lab for helpful discussions and feedback on later versions of the work. AG is supported by the NSF CSGrad4US Fellowship.

References

N. S. Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Donald T Campbell and Donald W Fiske. 1959. Convergent and discriminant validation by the multitrait-multimethod matrix. *Psychological bulletin*, 56(2):81.

Emmanuel J. Candes and Benjamin Recht. 2009. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772.

Samprit Chatterjee and Ali S. Hadi. 1986. Influential observations, high leverage points, and outliers in linear regression. *Statistical Science*, 1:379–393.

Harshita Diddee and Daphne Ippolito. 2025. Chasing random: Instruction selection strategies fail to generalize. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1943–1957, Albuquerque, New Mexico. Association for Computational Linguistics.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd edition. Springer.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300.

Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, Abir De, and Rishabh K. Iyer. 2021. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*.

David D. Lewis and Jason Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *International Conference on Machine Learning*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Alexander Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew Arad Hudson, and 31 others. 2023. Holistic evaluation of language models. *Transactions on Machine Learning Research*. Featured Certification, Expert Certification.

Ian Magnusson, Nguyen Tai, Ben Bogin, David Heineman, Jena Hwang, Luca Soldaini, Akshita Bhagia, Jiacheng Liu, Dirk Groeneveld, Oyvind Tafjord, Noah A. Smith, Pang Wei Koh, and Jesse Dodge. 2025. DataDecide: How to Predict Best Pretraining Data with Small Experiments. *arXiv preprint*.

Colin L Mallows. 1972. A note on asymptotic joint normality. *The Annals of Mathematical Statistics*, pages 508–515.

J.I. Marcum. 1960. A statistical theory of target detection by pulsed radar. *IRE Transactions on Information Theory*, 6(2):59–267.

Brian B. Moser, Arundhati S. Shanbhag, Stanislav Frolov, Federico Raue, Joachim Folz, and Andreas Dengel. 2025. A coreset selection of coreset selection literature: Introduction and recent advances. *ArXiv*, abs/2505.17799.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. 2021. Deep learning on a data diet: Finding important examples early in training. In *Neural Information Processing Systems*.

Yotam Perlitz, Elron Bandel, Ariel Gera, Ofir Arviv, Liat Ein-Dor, Eyal Shnarch, Noam Slonim, Michal Shmueli-Scheuer, and Leshem Choshen. 2024. Efficient benchmarking (of language models). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2519–2536, Mexico City, Mexico. Association for Computational Linguistics.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. 2024. tinybenchmarks: evaluating llms with fewer examples. *ArXiv*, abs/2402.14992.

Pedro Rodriguez, Joe Barrow, Alexander Miserlis Hoyle, John P. Lalor, Robin Jia, and Jordan Boyd-Graber. 2021. Evaluation examples are not equally informative: How should that change NLP leaderboards? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4486–4503, Online. Association for Computational Linguistics.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408.

Walter Rudin. 1987. *Real and complex analysis*. McGraw-Hill, Inc.

Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. 2021. Scaling up influence functions. In *AAAI Conference on Artificial Intelligence*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, and 431 others. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *ArXiv*, abs/2206.04615.

Nishant Subramani, Jason Eisner, Justin Svegliato, Benjamin Van Durme, Yu Su, and Sam Thomson. 2025. MICE for CATs: Model-internal confidence estimation for calibrating agents with tools. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 12362–12375, Albuquerque, New Mexico. Association for Computational Linguistics.

Rajan Vivek, Kawin Ethayarajh, Diyi Yang, and Douwe Kiela. 2023. Anchor points: Benchmarking models with much fewer examples. *ArXiv*, abs/2309.08638.

Ziang Xiao, Susu Zhang, Vivian Lai, and Q. Vera Liao. 2023. Evaluating evaluation metrics: A framework for analyzing NLG evaluation metrics using measurement theory. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10967–10982, Singapore. Association for Computational Linguistics.

Qinyuan Ye, Harvey Fu, Xiang Ren, and Robin Jia. 2023. How predictable are large language model capabilities? a case study on BIG-bench. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7493–7517, Singapore. Association for Computational Linguistics.

Vilém Zouhar, Peng Cui, and Mrinmaya Sachan. 2025. How to select datapoints for efficient human evaluation of nlg models? *ArXiv*, abs/2501.18251.

A Dataset Similarity Metrics

Here are more details about the dataset similarity metrics used in our analysis. Consider a pair of datasets (D_i, D_j); we use seven similarity measures SIM.

PEARSON CORRELATION: Measures linear relationships between dataset performance vectors but is sensitive to outliers and assumes linearity. We compute the PEARSON CORRELATION:

$$\rho_{D_i,D_j} = \frac{\sum(R_{m,D_i} - \bar{R}_{D_i})(R_{m,D_j} - \bar{R}_{D_j})}{\sqrt{\sum(R_{m,D_i} - \bar{R}_{D_i})^2 \sum(R_{m,D_j} - \bar{R}_{D_j})^2}} \tag{9}$$

where R_{m,D_i} is the performance of model m on dataset D_i and $\bar{R}_{D_i}$ is the average performance across all models for dataset D_i,

SPEARMAN CORRELATION: A ranked correlation that handles non-linear monotonic relationships but is sensitive to small perturbations that flip ranks.

$$\rho_s = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{10}$$

where d_i is the rank difference between model performances on datasets D_i and D_j,

KENDALL-TAU CORRELATION (KENDALL, 1938): Another ranked correlation that measures agreement in the orderings of data but also sensitive to small perturbations.

$$\tau = \frac{C - D}{C + D} \tag{11}$$

where C represents concordant model rankings across two datasets, and D represents discordant rankings.

COSINE SIMILARITY: Measures the cosine of the angle between performance vectors, capturing directional similarity regardless of magnitude differences between datasets.

$$\text{COSINE\_SIMILARITY}(D_i, D_j) = \frac{B[:,i]B[:,j]}{||B[:,i]||||B[:,j]||} \tag{12}$$

LP NORM (RUDIN, 1987) BASED SIMILARITIES: We define a family of similarity measures based on Lp norms with exponential normalization:

$$\text{LP\_SIMILARITY}(D_i, D_j) = \exp\left(-||B[:,i] - B[:,j]||_p\right) \tag{13}$$

We employed various distance-based similarity measures using exponential normalization to capture different aspects of performance similarity between datasets. We specifically use L1 (MANHATTAN), L2 (EUCLIDEAN), and L3 (MINKOWSKI) norms. The exponential normalization ensures all similarities are bounded in (0,1], with identical performance patterns yielding similarity 1 and increasingly dissimilar patterns approaching 0.

WASSERSTEIN SIMILARITY (MALLOWS, 1972): Measures the minimum "cost" of transforming one performance distribution into another, capturing both shape and statistical differences. We also exponentially normalize this such that similarities are bounded in (0, 1].

$$\text{WASSERSTEIN\_SIMILARITY}(D_i, D_j) = \exp\left(\frac{-W_1(B[:,i], B[:,j])}{\max_{k,l} W_1(B[:,k], B[:,l])}\right) \tag{14}$$

JENSEN-SHANNON SIMILARITY:

$$\text{JENSEN\_SHANNON\_SIMILARITY}(D_i, D_j) = 1 - \sqrt{\frac{D_{KL}(P_i||M) + D_{KL}(P_j||M)}{2}} \tag{15}$$

Here $M = \frac{1}{2}(P_i + P_j)$ and P_i, P_j are normalized distributions of $B[:,i], B[:,j]$. Jensen-Shannon similarity provides a symmetric measure based on information theory that quantifies dataset distributional differences.

B Proportions Better than Random

To measure how well each system (other than random) fares across the 1000 random runs of the RANDOM baseline, we measure the proportion of the 1000 random runs that each system does as well or better than. We measure this across two metrics on all three benchmarks. The first metric, "AUC", is measured by SCAUC. The second metric, "Max2", looks at the SCAUC up until a representative dataset S is discovered that achieves at least 95% coverage (S*). This is compared to the RANDOM baseline for the same number of datasets (|S*| under a similarity function SIM) across all 1000 runs. Note that SCAUC cannot be computed if |S*|=1, so if |S*|=1, we consider the first two datasets here. These results are below in Table 6 as proportions. We find that GREEDY MINIMUM and GREEDY MAXIMUM both perform worse than RANDOM. On MMLU, all similarity measures outperform RANDOM, but on HELM and BigBenchLite, only about half the systems outperform RANDOM on SCAUC on average. Our representative dataset discovery algorithm generally outperforms RANDOM early on until S* is discovered regardless of similarity function, with most systems strongly outperforming RANDOM on "Max2."

	HELM		MMLU		BigBenchLite	
Similarity Methods	AUC (↑)	Max2 (↑)	AUC (↑)	Max2 (↑)	AUC (↑)	Max2 (↑)
RANDOM (baseline)	–	–	–	–	–	–
GREEDY MINIMUM	0.095	0.095	0.000	0.000	0.000	0.000
GREEDY MAXIMUM	0.030	0.028	0.002	0.223	0.521	0.504
PEARSON	0.626	0.970	**0.972**	**1.000**	0.068	0.086
SPEARMAN	0.213	0.415	0.108	0.994	0.057	0.072
KENDALL-TAU	0.552	0.970	0.330	0.994	0.141	0.154
COSINE	0.427	0.409	0.124	0.880	0.115	0.132
MANHATTAN (L1)	**0.685**	0.501	0.825	0.924	0.727	0.849
EUCLIDEAN	0.595	0.944	0.931	0.868	0.692	0.832
MINKOWSKI (L3)	0.538	**0.973**	0.939	0.965	**0.815**	**0.928**
WASSERSTEIN	0.581	0.507	0.962	0.965	0.692	0.791
JENSEN-SHANNON	0.377	0.409	0.060	0.919	0.155	0.303

Table 6: Proportion of methods that perform better than or equal to RANDOM across three evaluation metrics. Values closer to 1.0 indicate better performance relative to RANDOM. **Bold** indicates the best performing method for each metric within each dataset.