Efficient Hyperparameter Tuning of the κ-η Regression Method by Training Data Sampling

Kevin Baboolal
Dept. of Computer Science
The University of the West Indies
St. Augustine, Trinidad and Tobago
kevin@lab.tt

Patrick Hosein
Dept. of Electrical and Computer Engineering
The University of the West Indies
St. Augustine, Trinidad and Tobago
patrick.hosein@uwi.edu

***Abstract*—The κ-η regression method, presents a significant computational challenge for hyperparameter optimization due to its inherent quadratic time complexity for training, ($\mathcal{O}(N^2)$), which renders exhaustive grid search on large datasets prohibitive. This paper investigates sampling strategies as a means to mitigate this complexity. We evaluate the efficacy of performing hyperparameter tuning on a reduced subset (25%) of the training data using two distinct methods: uniform random sampling and Locality-Sensitive Hashing (LSH). Our empirical analysis, conducted across eight UCI regression datasets, demonstrates that both sampling techniques yield substantial reductions in computation time, ranging from 22% to 82%, compared to a full grid search. LSH-based sampling shows consistent performance in estimating the hyperparameters of the algorithm. The performance of LSH is attributed to its ability to generate a representative subsample that preserves the local geometric structure of the data, a property to which the distance-based κ-η regressor is highly sensitive. The findings validate LSH-based sampling as a robust and efficient methodology for optimizing the κ-η algorithm, offering a compelling trade-off between predictive accuracy and computational cost.**

***Index Terms*—Sampling, Locality-Sensitive Hashing, Search Optimization**

I. Introduction

Hyperparameter optimization remains a fundamental challenge in machine learning, particularly for models with multiple interdependent parameters. Traditional grid search approaches, while thorough, become computationally prohibitive as the parameter space grows. This problem occurs in non-parametric, distance-based models where hyperparameters directly influence the weighting mechanisms used for predictions. As the parameter space grows the number of permutations grow significantly. The κ-η algorithm benefits from having two hyperparameters, however it's complexity for training grows in $\mathcal{O}(N^2)$. The hyperparameters control the decay rate of influence based on distance, making it critical to model performance. The challenge lies in efficiently optimizing multiple parameters while simultaneously while maintaining predictive accuracy.

This work contributes to the field in two key areas: (1) we provide an evaluation of sampling strategies for hyperparameter optimization in the κ-η algorithm , and (2) we demonstrate that LSH-based or random sampling can achieve near-optimal performance with substantial computational savings. Next we will formally introduce the κ-η algorithm

978-1-6654-7808-3/25/$31.00 ©2025 IEEE

II. κ-η Algorithm

The algorithm originated from explorations into the personalization versus bias trade-off in predictive tasks, particularly for automobile insurance claims [1]. The method is a supervised learning method designed for regression tasks, and functions via two primary stages, both stages depend on the kernel $\frac{1}{(1+d)^\theta}$ where d represents the distance between two points and θ is the weighting hyperparameter (κ or η). The κ-η algorithm uniquely employs a distance-based weighting mechanism to first encode the features and subsequently predict a target value. During the first stage we embed each feature into the target space using the kernel and we call this step κ-Encoding. Next we create a prediction using the same kernel, this step is referred to as η-Regression. The following section looks at both steps in more detail.

A. κ-Encoding

In the initial stage, the algorithm transforms raw feature values into a new, encoded representation. For every feature, k and each sample i (whether from the training set S or the test set T), the encoder computes a new value, $\hat{x}_i[k]$. This encoded value is formulated as a weighted average of the target values y_j from all samples j within the training set S. The influence of each training sample is weighted inversely to the distance between the feature values $x_i[k]$ and $x_j[k]$, a process modulated by the hyperparameter $\kappa \geq 0$.

The specific encoding formula for feature, k of sample, i is expressed as:

$$\hat{x}_i[k] = \frac{\sum_{j \in S} \frac{y_j}{(1+|x_i[k]-x_j[k]|)^\kappa}}{\sum_{j \in S} \frac{1}{(1+|x_i[k]-x_j[k]|)^\kappa}} \tag{1}$$

In the case where $\kappa = 0$, the encoded value $\hat{x}_i[k]$ simplifies to the mean of the target values across the training set. Conversely, as $\kappa[k] \to \infty$, the encoding converges towards the average target value of only those training samples that have the exact same feature value as sample i. This encoding procedure is applied to every feature to create a fully encoded dataset.

B. η-Regressor

The second stage utilizes the newly encoded dataset to generate predictions. For a given test sample t, the regressor estimates its target value $\hat{y}_t$ by calculating a weighted average of the target values y_i of all training samples $i \in S$. The weights in this stage are determined by the Euclidean distance D_{ti} between the complete encoded feature vectors of the test sample t and each training sample i. The hyperparameter, $\eta \geq 0$, governs this regression stage.

Mathematically, the Euclidean distance between the encoded features of the test sample t and a training sample i is found by:

$$D_{ti} = \sqrt{\sum_{k \in F} (\hat{x}_t[k] - \hat{x}_i[k])^2}$$

Next, the final prediction $\hat{y}_t$ is calculated using these distances:

$$\hat{y}_t = \frac{\sum_{i \in S} \frac{y_i}{(1+D_{ti})^\eta}}{\sum_{i \in S} \frac{1}{(1+D_{ti})^\eta}} \tag{2}$$

C. Time and Space Complexity

Let $N = |S|$ represent the number of samples in the training set, $M = |T|$ be the number of samples in the test set, and F denote the number of features.

1) Time Complexity: The total time complexity is an aggregation of the complexities of both the encoding and regression stages.

- κ-Encoder: To encode a single feature across all M samples, the algorithm must compute a weighted sum over all N training points for each unique value of that feature. In a worst-case scenario where the number of unique values is on the order of N, the complexity to encode all F features becomes $\mathcal{O}(F \cdot N^2)$.
- η-Regression: For each of the M test samples, it calculates the Euclidean distance to all N training samples (an $\mathcal{O}(F)$ operation for each distance), and then computes the final weighted average. This leads to a time complexity of $\mathcal{O}(M \cdot N \cdot F)$.

In a standard hyperparameter tuning process, such as k-fold cross-validation, the training and test set sizes are fractions of N. The most time-consuming part of each evaluation is either the encoder's transform operation ($\mathcal{O}(F \cdot N^2)$) or the regressor's prediction ($\mathcal{O}(N \cdot N \cdot F) = \mathcal{O}(F \cdot N^2)$) for $N = M$. Consequently, the total time for a comprehensive hyperparameter search is:

$$\mathcal{O}(\text{num\_folds} \cdot \text{num\_hyperparams} \cdot F \cdot N^2)$$

This quadratic relationship with the number of training samples, N, can render hyperparameter tuning long for large datasets.

2) Space Complexity: The algorithm's memory requirement is dictated by the data stored during execution.

- κ-Encoder: This stage needs to hold the original training features and target values, which requires $\mathcal{O}(N \cdot F)$ space.
- η-Regressor: Similarly, this stage must store the encoded training features and the corresponding target values, also demanding $\mathcal{O}(N \cdot F)$ space.

The total space complexity is therefore $\mathcal{O}(N \cdot F)$, scaling linearly with the size of the training data.

III. Motivation - Effect of Sample Size on Predicting Optimal Hyperparameters

The κ-η algorithm is a data dependent kernel making the hyperparameter selection highly influenced by the data used for training. Consequently, the hyperparameter value optimized on a larger sample is more likely to be the true optimal value for the complete dataset. To quantify this effect, we varied the sample size for hyperparameter selection and observed the MSE on a hold out set. We repeat this experiment 50 times and the results are depicted in Figure 1, revealing two critical trends. Firstly, the mean MSE exhibits a general decrease as the sample size grows tending towards the true MSE when using the hyper-parameters selected with the full training dataset.

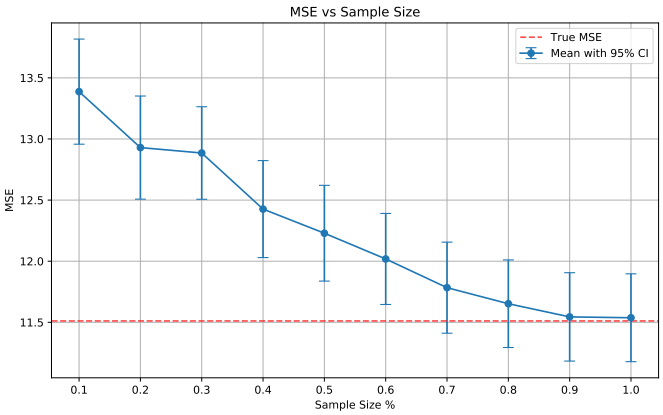

Fig. 1. Depicting the change in MSE at different sampling rates

Secondly, the confidence interval width (Table I) shows a general downward trend, decreasing from 0.86 to 0.72. Wider intervals at smaller sample sizes reflects greater uncertainty and variability in the performance estimate. As more data is included in the sample, our estimate becomes more precise, and the confidence interval tightens. This narrowing signifies increased confidence that the observed MSE on the sample is close to the true MSE.

The experiment reveals that while using smaller samples for hyperparameter tuning offers significant computational advantages, it comes with the risk of selecting a suboptimal hyperparameter due to sample-specific noise and a less accurate representation of the true data distribution. This raises the opportunity for creating a robust sampling method for the

TABLE I
IMPACT OF INCREASING SAMPLE SIZE ON MSE ESTIMATION

Sample Size	Lower Bound	Upper Bound	Width	MSE	Δ True MSE
10%	12.96	13.82	0.86	13.39	-1.88
20%	12.51	13.35	0.84	12.93	-1.42
30%	12.51	13.26	0.76	12.89	-1.37
40%	12.03	12.82	0.79	12.43	-0.91
50%	11.84	12.62	0.78	12.23	-0.72
60%	11.65	12.39	0.74	12.02	-0.51
70%	11.41	12.16	0.75	11.78	-0.27
80%	11.29	12.01	0.72	11.65	-0.14
90%	11.18	11.91	0.72	11.54	-0.03
100%	11.18	11.90	0.72	11.54	-0.00

κ-η Algorithm that approximates reliable hyperparameters with lowered computational cost

IV. METHODOLOGY

Given the κ-η algorithm's quadratic time complexity, tuning hyperparameters can become a bottleneck on large datasets. Rather than using the entire dataset of N samples for the hyperparameter search, one can perform the search on a smaller, yet representative, subset of size $N_{subset} \ll N$. Evaluating the performance of N_{subset} we select samples using random sampling and locality sensitive hashing (LSH). We also run hyperparameter selection using the full training data to determine the chnage in performance. We next describe the implementation of the aforementioned methods.

A. Full training data grid search

The baseline approach performs exhaustive search over all parameter combinations using 3-fold cross-validation and the full training data. For κ-encoder we search $\kappa \in [0, 80]$ (step 2) and η-regressor $\eta \in [0, 80]$ (step 2).

B. Random Sampling

Random sampling selects 25% of the training data uniformly at random for hyperparameter optimization. This approach maintains the same search parameter space but reduces the computational cost through data reduction.

C. Locality Sensitivity Hashing for Sampling

Locality Sensitive Hashing [2] is a method for efficiently finding approximate nearest neighbors in high-dimensional spaces. The fundamental principle is to use a family of hash functions $\mathcal{H}$ such that for any two points p, q:

- If p and q are *close*, the probability of them hashing to the same value (colliding) is high.
- If p and q are *far*, the probability of them colliding is low.

This allows us to get representative samples based on data point collisions. We implement the random projection method as follows:

1) The Hash Function: A single hash function $h(v)$ is defined by a single random hyperplane. This hyperplane is itself defined by its random normal vector r. The hash function simply determines on which side of the hyperplane a given data vector $v \in \mathbb{R}^d$ lies. This is calculated using the dot product:

$$h_r(v) = \begin{cases} 1 & \text{if } r \cdot v \geq 0 \\ 0 & \text{if } r \cdot v < 0 \end{cases}$$

In essence, the hyperplane partitions the entire vector space into two halves, and the hash function assigns a single bit (1 or 0) to each vector.

A single hash function is not discriminative enough. To improve performance, we use a strategy involving two parameters, k and L.

1) We create a single hash table by concatenating k independent random projection hash functions $\{h_1, \ldots, h_k\}$. A hash "bucket" is now identified by a k-bit signature:
$$g(v) = (h_1(v), h_2(v), \ldots, h_k(v))$$
2) We repeat the Concatenation construction L times, creating L independent hash tables, each with its own set of k hash functions $\{g_1, \ldots, g_L\}$. Two vectors u and v are considered a candidate pair if they collide in *at least one* of these L tables.

By tuning k and L, we can separate close neighbors from distant points, enabling an efficient search. If we denote the number of samples by C then we select k and L using heuristic values as follows:

$$k, \text{ hyperplanes} = \lceil \log_2(C) \rceil$$
$$L, \text{ tables} = \lceil \ln(C) \rceil$$

We then perform sampling across buckets in a round robin fashion until we have selected 25% of the dataset. This approach aims to select samples that span the feature space more effectively than pure random sampling, potentially capturing important boundary regions and representative clusters within the data distribution.

D. Performance Evaluation

Sampling performance is evaluated using Mean Squared Error (MSE) on held-out test sets. We employ a nested validation strategy:

1) Split data into train (80%) and test (20%)
2) We sample 25% of the training data for hyperparameter optimization.
3) Perform 3-fold cross-validation with the selected samples to find the optimal parameters.
4) Evaluate final model on the test set with optimal parameters selected.

E. Datasets

We conduct experiments on eight regression datasets from the UCI Machine Learning [3], spanning diverse domains and characteristics as shown in Table II. All datasets are preprocessed including missing value imputation and Min-Max scaling to ensure fair comparison across methods.

TABLE II
DATASET STATISTICS

Dataset	Samples	Features	Domain
Horse Colic	3,196	36	Medical
Airfoil Self-Noise	1,503	5	Engineering
Auto MPG	398	7	Automotive
Automobile	205	60	Automotive
Auto Univ	625	4	Automotive
CPS	699	9	Economics
Facebook Metrics	500	18	Business

F. Implementation Details

Our implementation leverages Numba [4] for high-performance numerical computations, particularly in distance calculations and weight computations. All experiments use a fixed random seed (42) for reproducibility. The hyperparameter search grid spans values from 0 to 80 with step size 2 for both parameteers κ and η, providing a comprehensive search space while maintaining computational feasibility. All code to implement and reproduce results are available at [5].

V. RESULTS AND ANALYSIS

Table III presents a comprehensive summary of the experimental results on eight datasets. The results consistently demonstrate that sampling-based approaches for hyperparameter optimization offer a compelling trade-off between computational efficiency and predictive accuracy. For each dataset, we report the execution time, the optimal hyperparameters (κ and η) identified, the resulting test Mean Squared Error (MSE), the percentage reduction in computation time, and both the absolute and relative change in MSE compared to the full dataset baseline.

A. Aggregate Performance Analysis

Aggregating the results across all datasets reveals several key insights into the effectiveness of the sampling strategies. The comprehensive analysis highlights the following findings:

1) Viable Trade-Off: Both Random and LSH sampling achieve substantial reductions in computation time, ranging from 22% to over 82%, while incurring only a modest increase in test MSE in most cases. This demonstrates that sampling is a highly effective strategy for accelerating hyperparameter tuning.
2) Superiority of LSH: In 7 out of 8 datasets, LSH sampling resulted in a lower test MSE compared to Random sampling. On datasets like *Auto MPG* and *Auto Univ*, LSH achieved a performance degradation of less than 1%, far superior to the 10.2% and 4.8% degradation from Random sampling, respectively. This suggests that LSH's ability to preserve data locality creates more representative subsamples.

B. Computational Efficiency Analysis

The computational benefits become more pronounced as dataset complexity and size increase. We plot the sampling method against dataset size and search time in Figure 2. All curves display a quadratic relationship, however, the sampled methods are order of magnitudes less. The Time Reduction column in Table III quantifies this efficiency gain. For instance, the hyperparameter search on the *Airfoil Self-Noise* and *Facebook Metrics* datasets were accelerated by approximately 64-65%. This dramatic reduction in runtime allows for more extensive hyperparameter searches or faster model development cycles, which is a crucial advantage in practical machine learning applications where computational resources and time are often constrained. LSH and Random sampling offer nearly identical time reductions.

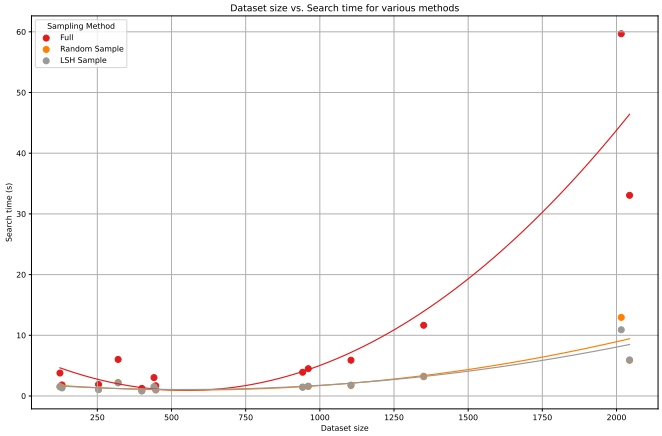

Fig. 2. Time savings of the Sampling Approaches

C. Parameter Selection Analysis

The optimal κ and η values found using sampling often differ from those identified by the full search. This suggests that the hyperparameter space is complex, and that sampling explores different regions, sometimes finding equally effective or even better-generalizing parameter combinations. LSH consistently identifies parameters that yield a test MSE very close to the baseline, as seen on *Auto MPG* (Δ MSE % of 0.5%) and *Auto Univ* (0.6%). This demonstrates that even though the specific parameter values may differ, LSH robustly finds subsamples that lead to models with comparable predictive power to those trained on the complete data. The effectiveness of sampling suggests that it is not always necessary to exhaustively search the entire parameter grid on the full dataset to find a high-performing model.

VI. DISCUSSION

The empirical results advocate for sampling-based hyperparameter optimization, with a notable performance advantage for Locality-Sensitive Hashing (LSH). The following sections discuss these findings.

A. LSH and the κ-η -Regressor

The κ-η Regressor is a distance-based, non-parametric learner, where the prediction for a test point x_t is a weighted average of all training targets y_i, formally:

TABLE III
EXPERIMENTAL RESULTS

Dataset	Method	Training Size	Time (s)	Optimal κ/η	Test MSE	Time Reduction	Δ MSE	Δ MSE %
Airfoil Self-Noise	Full	961	4.50	2/32	10.2006	–	–	–
	Random	241	1.57	4/16	11.9579	65.1%	1.757	17.2%
	LSH	240	1.62	2/20	11.2336	64.0%	1.033	10.1%
Auto MPG	Full	254	1.90	2/6	4.5676	–	–	–
	Random	64	1.06	10/8	5.0354	44.2%	0.468	10.2%
	LSH	63	1.05	16/4	4.5915	44.7%	0.024	0.5%
Auto Univ	Full	400	1.25	2/46	0.2621	–	–	–
	Random	100	0.86	2/32	0.2747	31.2%	0.013	4.8%
	LSH	100	0.82	2/42	0.2638	34.4%	0.002	0.6%
Automobile	Full	131	1.82	8/18	0.0378	–	–	–
	Random	33	1.42	22/16	0.0442	22.0%	0.006	16.9%
	LSH	32	1.32	16/20	0.0389	27.5%	0.001	2.9%
CPS	Full	447	1.69	32/8	0.1013	–	–	–
	Random	112	1.01	14/20	0.1356	40.2%	0.034	33.9%
	LSH	111	1.00	26/12	0.1127	40.8%	0.011	11.3%
Facebook Metrics	Full	320	6.02	78/12	1244	–	–	–
	Random	80	2.22	78/10	1225	63.1%	-18.341	-1.5%
	LSH	80	2.15	78/10	1225	64.3%	-18.341	-1.5%
Horse Colic	Full	2,044	33.1	34/34	0.0291	–	–	–
	Random	511	5.88	4/34	0.0318	82.2%	0.003	9.3%
	LSH	511	5.94	4/36	0.0315	82.0%	0.002	8.2%
Pharynx	Full	441	3.03	12/18	0.0434	–	–	–
	Random	111	1.51	12/38	0.0395	50.2%	-0.004	-9.0%
	LSH	110	1.50	20/50	0.0410	50.5%	-0.002	-5.5%

$$\hat{y}(x_t) = \frac{\sum_{i=1}^{N_{train}} y_i \cdot w_i}{\sum_{i=1}^{N_{train}} w_i}, \quad \text{where} \quad w_i = (1 + \|x_t - x_i\|_2)^{-\kappa}$$

This formulation shows that the prediction is highly sensitive to the local neighborhood structure of the training data. Points x_i closer to x_t contribute more to the final prediction. Random sampling, by its uniform nature, is agnostic to the structure. This can mislead the hyperparameter search into selecting suboptimal parameters that, for instance, over-smooths in a region that has been sparsified. LSH, in contrast, is designed to approximate a nearest-neighbor search. By hashing points such that nearby points in the Euclidean space are likely to collide in the same bucket, LSH-based sampling attempts to preserve local neighborhoods. This allows the cross-validation process to obtain a more accurate estimate of the true error, leading to the selection of more robust and effective hyperparameters, as evidenced by Table III.

B. Scalability and Algorithmic Complexity

The practical implications of this research are most evident when analyzing the computational complexity. A full grid search for hyperparameters requires training and validating the model for each point in the parameter grid. This process has a complexity of $O(N_{test} \cdot N_{train} \cdot D)$, where D is the number of features. Sampling directly reduces the dominant N_{train} term to N_{sample}. This yields a theoretical speedup factor proportional to $(N_{train}/N_{sample})^2$ inline with the quadratic relationship observed in Figure 2.

VII. RELATED WORKS

A. κ-η Regression

The initial work done by [1] proposed a model to predict annual claim rates by employing a weighted average of historical claim rates that have been embedded into the target space. The weighting mechanism is inversely proportional to the distance between a test sample and training samples, raised to the power of a hyperparameter, κ. This κ parameter is fundamental, as it controls the balance between high personalization (large κ, focusing on very close samples) and robustness (small κ, averaging over a wider neighborhood) [1]. The concept defines distances between categorical and numerical features by mapping them to the target variable, such as average claim rates [6]. The κ-regression framework was redefined into a two-stage process to enhance its explainability, particularly for multivariate datasets. The first stage, termed κ-encoding, was formalized as target encoding method applicable to both categorical and continuous data [7]. In this stage, raw feature values are transformed into an encoded representation by computing a weighted average of target values based on

feature distances, again modulated by a κ hyperparameter [7]. This ensures all features are comparable by embedding them into a common target space. Subsequently, the second stage, now referred to as η-regression (or the predictive phase), utilizes these κ-encoded features to generate final predictions, applying a second weighted average of target values, governed by a hyperparameter η. The comprehensive performance evaluation of this two-parameter regression algorithm against conventional machine learning models confirmed its high accuracy and robustness [8]. This work explicitly identified the two hyper-parameters as, κ_1 for encoding and κ_2 for prediction.

A significant challenge identified early in the development of the regression framework was the computational intensity associated with hyperparameter optimization [6]. The base κ-η algorithm exhibits a time complexity of $O(F \cdot N^2)$ for both the encoding and regression stages, where F is the number of features and N is the number of samples. To mitigate this, several efficient hyperparameter tuning approaches were explored. One method proposed a Successive Quadratic Approximation (SQA) approach, leveraging the observed (initially convex, then concave) shape of the error function with respect to κ [9]. This iterative method demonstrated significant speedups compared to a linear grid search. Another approach introduced a modified Golden Section Search algorithm, capitalizing on the experimentally observed nature of the prediction error as a function of the parameter value [10]. This method efficiently narrowed down the search interval by statistically estimating the optimal κ value's range and using an error tolerance stopping condition, leading to substantial reductions in iteration count compared to linear search [10].

The underlying distance-based weighting mechanism, central to the κ-η framework, has also been explored in other machine learning contexts. A similar kernel was applied to Bayes classification, demonstrating an ability to estimate conditional probabilities using neighborhood information and addressing the zero-frequency problem without assuming attribute independence. [11]

VIII. Limitations and Future Research

Despite the promising results, the cross-validation error on a subsample, is a biased estimator of the true generalization error. This bias is apparent in datasets like *Airfoil Self-Noise*, where sampling led to a notable performance change. The efficacy of LSH is contingent upon its own set of hyperparameters (e.g., hash function width, number of hash tables). This study did not optimize these meta-parameters, presenting a potential area for further performance gains but also introducing another layer of model selection complexity. Building on these observations, future research could explore for a given test point, using LSH index to query the approximate nearest neighbors. Allowing the regressor to mutate parameters based on the density of the test point. One should also investigate other stratified sampling techniques that bin the data based on quantiles of the target variable. Comparing a target-aware strategy to the feature-aware LSH could reveal deeper insights into the algorithm's behavior. Sampling resulted in better MSE than the full search method suggesting an opportunity for adding regularization to the model in the future.

IX. Conclusion

This study demonstrates that intelligent sampling strategies can significantly improve the efficiency of hyperparameter optimization for κ-η regression models without sacrificing predictive performance. Through comprehensive evaluation on 8 diverse UCI datasets, we show that LSH-based sampling or random sampling reduces hyperparameter optimization time in the range of 22%-82% with particularly strong benefits for larger datasets. LSH-based sampling shows strong performance in picking reliable parameters with heuristics ($\lceil \log_2(n) \rceil$ hyperplanes, $\lceil \ln(n) \rceil$ tables) eliminating manual tuning while maintaining effectiveness across diverse dataset characteristics. The success of this approach highlights the importance of geometric structure preservation in hyperparameter optimization for distance-based models. For Data Scientists working with κ-η regression or similar distance-based models, our results suggest that LSH-based sampling can provide an excellent balance between computational efficiency and model performance, particularly for datasets with moderate to large sample sizes.

References

[1] P. Hosein, "On the prediction of automobile insurance claims: The personalization versus confidence trade-off," in *2021 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD)*, 2021, pp. 1–6.

[2] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases*, ser. VLDB '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, p. 518–529.

[3] K. N. Markelle Kelly, Rachel Longjohn, "UCI machine learning repository." [Online]. Available: https://archive.ics.uci.edu

[4] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: a llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, ser. LLVM '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2833157.2833162

[5] K. Baboolal, "Sampling experiments," 2025, gitHub repository. [Online]. Available: https://github.com/kb-papers/ictmod-25

[6] P. Hosein, "A data science approach to risk assessment for automobile insurance policies," 2023. [Online]. Available: https://doi.org/10.1007/s41060-023-00392-x

[7] K. Baboolal, S. Gooljar, and P. Hosein, "A novel approach to feature encoding," in *2023 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD)*, 2023, pp. 1–6.

[8] S. Gooljar, K. Manohar, and P. Hosein, "Performance evaluation and comparison of a new regression algorithm," in *Proceedings of the 12th International Conference on Data Science, Technology and Applications - DATA*, INSTICC. SciTePress, 2023, pp. 524–531.

[9] P. Hosein, K. Manohar, and K. Manohar, "A successive quadratic approximation approach for tuning parameters in a previously proposed regression algorithm," in *Proceedings of the 12th International Conference on Data Science, Technology and Applications - Volume 1: DATA*, INSTICC. SciTePress, 2023, pp. 629–633.

[10] D. Ramsubhag, K. Baboolal, and P. Hosein, "Efficient hyper-parameter tuning for the kappa regression algorithm," in *2024 International Conference on Decision Aid Sciences and Applications (DASA)*, 2024, pp. 1–6.

[11] P. Hosein and K. Baboolal, "Bayes classification using an approximation to the joint probability distribution of the attributes," in *Deep Learning Theory and Applications*, A. Fred, A. Hadjali, O. Gusikhin, and C. Sansone, Eds. Cham: Springer Nature Switzerland, 2024, pp. 47–61.