Fine-Tuning Machine Learning Regressors for Optimal Hyperparameter Selection

Omid Saberi
Dept. of Industrial Automation
University West
Trollhättan, Sweden
https://orcid.org/0009-0005-3596-9182

***Abstract*—This report investigates the fine-tuning of four machine learning regression models—Linear Regression, Decision Tree Regression, Random Forest Regression, and Support Vector Machine Regression—through the application of hyperparameter optimization techniques, including Randomized Search and Grid Search with 5-fold cross-validation. The objective is to enhance predictive accuracy and generalization by systematically adjusting key hyperparameters. Our experiments reveal that Random Forest Regression consistently outperformed the other models, achieving the highest accuracy and demonstrating robustness across various datasets. Additionally, we explored Lasso Regression using LassoCV, which simplifies the hyperparameter search process by fitting models along a regularization path. The results indicate that effective hyperparameter tuning significantly impacts model performance, highlighting its importance in the machine learning pipeline. This work contributed to a deeper understanding of the trade-offs between model complexity and predictive power, emphasizing the need for rigorous model evaluation in practical applications. The comparative analysis of the models provides valuable insights for selecting appropriate regression techniques in real-world implementations.**

***Index Terms*—regression models, hyperparameter tuning, randomized search, grid search, Lasso, random forest, support vector machine, cross-validation, machine learning.**

I. Introduction

In recent years, machine learning has gained significant traction across various domains, driven by the increasing availability of data and advancements in computational power. This report focuses on fine-tuning four widely used regression models: Linear Regression, Decision Tree Regression, Random Forest Regression, and Support Vector Machine Regression. The performance of these models relies heavily on their hyperparameters, which dictate the model structure and learning process.

To enhance predictive accuracy, this study employs Randomized Search and Grid Search techniques with 5-fold cross-validation to explore the hyperparameter space for each model. Additionally, Lasso Regression is examined using LassoCV to streamline hyperparameter tuning. Through comparative analysis, this report aims to highlight the impact of hyperparameter optimization on model performance and provide insights for selecting suitable regression techniques in future applications.

II. Fine-Tuning ML Models

To fine-tune four regression models—Linear Regression, Decision Tree Regression, Random Forest Regression, and Support Vector Machine (SVM) Regression—I employed both Randomized Search and Grid Search with 5-fold cross-validation. The objective was to identify the optimal hyperparameters that minimize the Root Mean Squared Error (RMSE). I started with a Randomized Search to achieve an initial significant improvement, followed by an iterative Grid Search where feasible to further optimize the hyperparameter combinations. I acknowledge that I sometimes needed to run the Randomized Search multiple times with different ranges of values in the parameter distributions to find the optimal ranges that led to improvements in the RMSE, rather than negative effects.

A. Linear Regression

Linear Regression has fewer hyperparameters to tune compared to other models. However, I included cluster sizes from the preprocessing step in the parameter distribution to fine-tune the model. Additionally, I explored different variants of linear models, specifically Ridge Regression and Lasso Regression.

Ridge Regression adds an L2 penalty to the loss function, which helps to reduce overfitting by shrinking the coefficients of less important features. This is particularly useful in cases where the predictor variables are highly correlated [1]. Conversely, Lasso Regression incorporates an L1 penalty, which can lead to sparse solutions by forcing some coefficients to exactly zero, effectively performing variable selection. This can enhance model interpretability by retaining only the most significant predictors [2].

Ridge (L2 regularization) and Lasso (L1 regularization) regressions were tuned by optimizing the alpha hyperparameter. Alpha controls the strength of regularization, with higher values leading to increased regularization. For Ridge, this results in shrinkage of the coefficients, pushing them closer to zero [3], while Lasso performs both regularization and feature selection by driving some coefficients to exactly zero [2].

The hyperparameters of interest for both Ridge and Lasso Regression include:

- controls the strength of the regularization. A larger alpha increases the amount of shrinkage applied to the coefficients.
- Fit Intercept indicates whether to include an intercept in the model.

In my Randomized Search, I focused on tuning the alpha parameter, setting the parameter distributions to explore values ranging from 10^{-4} to 10^{2}, ensuring a comprehensive evaluation of the regularization strength across a wide range.

a) Random Search Results: During the randomized search for Lasso Regression, several issues and errors arose, resulting in the use of LassoCV, which handles the search internally by iteratively fitting along a regularization path. This approach yielded better results than the tuned simple linear regression. Before any tuning, the simple linear regression produced an RMSE of $71,648, as shown in Table I. By tuning the cluster size hyperparameter in the preprocessing pipeline with the same regressor, I achieved a slight reduction in error, resulting in 18 clusters for geographical categorization and an RMSE of $71,144—an improvement that was not particularly significant. In contrast, the LassoCV, as well as the Ridge regressors, significantly reduced the error, bringing RMSE down from $70,587 and $71,570 to $69,735 and $69,477, respectively. The hyperparameters contributing to the Ridge regressor's results were 14 for cluster sizes in the geographical preprocessing step and 193 for the alpha parameter in the Ridge regressor pipeline.

b) Grid Search Results: I performed the grid search only with the Ridge regressor, which improved the RMSE by an additional $553 in median housing value estimation. To initiate the grid search, I used the result from the random search as a baseline. Overall, the RMSE decreased from $71,570 to $68,944, reflecting a drop of $2,626 when utilizing both the randomized and grid search methods.

I selected the Ridge regressor to represent all the linear models in Fig. 2. Additionally, I provided a line graph in Fig. 1 that juxtaposes the three linear models for comparison.

B. Decision Tree Regression

The Decision Tree Regressor has several hyperparameters that significantly influence its performance the most important of which are as follows.

- max_depth determines the maximum depth of the tree. Limiting the depth can prevent overfitting, as deeper trees can capture noise in the training data. During our fine-tuning, we varied this parameter to find the optimal depth that balances bias and variance.
- min_samples_split is the minimum number of samples required to split an internal node. A higher value can lead to a more generalized model, while a lower value allows the model to create more splits and possibly overfit the data. We explored different values to ensure the tree did not become too complex.
- min_samples_leaf specifies the minimum number of samples that must be present in a leaf node. Setting this value helps in reducing overfitting by ensuring that leaf nodes contain a sufficient number of samples, leading to more stable predictions.
- max_features determines the number of features to consider when looking for the best split. We adjusted this parameter to explore whether considering fewer features could lead to better performance.

TABLE I
RMSE VALUES OF LINEAR REGRESSION MODELS AT DIFFERENT STAGES

Model	Before Fine-Tuning	After Random Search
Linear Regression	71,648	71,144
Lasso	70,587	69,735
Ridge	71,570	69,477

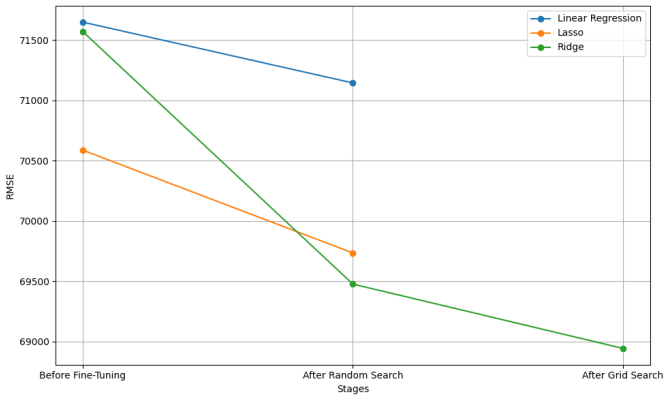

Fig. 1. RMSE of Linear Regression Models at Various Stages.

During the fine-tuning process, I utilized randomized search to optimize these hyperparameters, ultimately identifying values that enhanced the model's accuracy.

a) Randomized Search Results: Using randomized search, the model managed to significantly improve the result generated by the Decision Tree Regressor to 55,285, marking an improvement of approximately 10,800 units from the previous value of 66,074 before the fine-tuning process.

b) Grid Search Results: Starting from the base hyperparameters resulted form the randomized search, I achieved a slight improvement of 375 units by using the grid search method to fine-tune the Decision Tree Regressor. Table II displays the hyperparameters selected by the random and grid search methods at various stages of fine-tuning.

C. Random Forest Regression

To fine-tune the random forest regressor, I adjusted the same hyperparameters as those used with the decision tree regressor employing the random and grid search methods.

TABLE II
HYPERPARAMETERS SELECTED BY RANDOM AND GRID SEARCH

Method	max	feat.	min leaf	min split	clusters
Randomized	13	15	10	6	14
Grid	12	14	9	2	14

a) Random Search Results: With the randomized search method, RMSE fell from $47,140 to $43,261 by adjustments described in Table III.

b) Grid Search Results: The grid search would take too long to run given the number of hyperparameters I aimed to adjust for fine-tuning the random forest regressor. As a result, I found it difficult, if not impossible, to use the search method within a while loop to find even a locally optimal solution, as the method does not guarantee a global optimum. Therefore, I initially started with two values for each hyperparameter and, in the final stage, concluded the process by checking three values for each one in the parameters grid. The grid search provided an RMSE of $42,613.

D. Support Vector Machine Regression (SVR)

To fine-tune the SVR, I needed to provide a wider range for the C and epsilon hyperparameters. The C parameter is the regularization parameter, which controls the trade-off between achieving a low training error and a low testing error, essentially balancing underfitting and overfitting. A small C allows the model to tolerate more error, resulting in a smoother decision boundary but potentially higher bias. A large C reduces the tolerance for errors, leading to a more complex model that fits the training data more closely but risks overfitting. By exploring a wide range of C values, one ensures that they are not missing a better balance between bias and variance, which can significantly impact performance [3]. In Support Vector Regression (SVR), the epsilon-insensitive error function controls the tolerance for prediction errors, ignoring errors below a specified threshold epsilon. This allows the model to focus on points with larger residuals, which become the support vectors that influence the model's predictions. A smaller epsilon might work well if the data has subtle patterns that should be captured, while a larger epsilon could be beneficial if the data is noisy and the model needs to generalize better by ignoring small deviations. This wide exploration helps to ensure that the chosen epsilon reflects the true error tolerance that leads to the best generalization performance for the given dataset [4]. This is why I used a wider range of values for the epsilon hyperparameter too.

a) Random Search Results: The Support Vector Regressor had the highest RMSE compared to the other models, with an error of 118,192 units. Despite a significant error reduction through randomized search—lowering it by around $30,000—I couldn't get the RMSE below $89,000. This improvement was achieved by identifying a C value of 10, an epsilon of 10^{-4}, and a linear kernel as the optimal combination.

b) Grid Search Results: Building on the results from the randomized search, I further fine-tuned the Support Vector Regressor. The refined grid targeted small adjustments around these values, including C = 10.0, epsilon = 10^{-4}, and 7 clusters. Grid search results showed the optimal configuration with 8 clusters, C = 20.0, a degree of 4, and epsilon = 10^{-5}, achieving a final RMSE of 83,267, a noticeable improvement over the previous run.

TABLE III
HYPERPARAMETERS AND RMSE FOR RANDOM FOREST REGRESSOR

Stage	clusters	depth	features	min leaf	min split
Default	100	None	auto	1	2
Rnd. Search	18	24	9	2	4
Grid Search	20	18	6	2	2

III. MODEL SELECTION AND EVALUATION

The best RMSE of 42,613 units was achieved by fine-tuning the Random Forest Regressor. However, when fitting the model to the testing dataset, the RMSE increased to 49,455. As seen in Fig. 2, despite this rise, the Random Forest Regressor still outperformed all other regressors, even fine-tuned and fit on the training dataset.

IV. CONCLUSION

In conclusion, fine-tuning regression models using Randomized Search and Grid Search significantly enhanced their predictive performance. Among the models tested—Linear Regression (including Ridge and Lasso variants), Decision Tree Regression, Random Forest Regression, and Support Vector Machine Regression—Random Forest emerged as the top performer, achieving the lowest RMSE of 42,613 during training, with a slight increase to 49,455 on the test set. This indicates strong generalization capability despite the increase in error, which remains acceptable compared to other regressors.

The analysis highlits the importance of hyperparameter tuning, particularly for complex models like Random Forest and Support Vector Regression. Overall, while Random Forest yielded the most accurate predictions, each model provided valuable insights into the impact of hyperparameter tuning on regression outcomes, highlighting the critical role of model selection in practical applications.

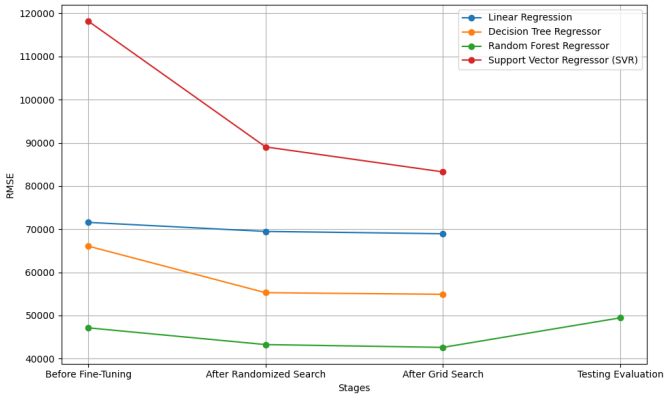

Fig. 2. RMSE of Different Regressors at Various Stages

REFERENCES

[1] A. E. Hoerl and R. W. Kennard, "Ridge regression: biased estimation for nonorthogonal problems," Technometrics, vol. 12, no. 1, pp. 55–67, 1970.

[2] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," Journal of the Royal Statistical Society. Series B (Methodological), vol. 58, no. 1, pp. 267–288, 1996.

[3] C. Cortes and V. Vapnik, "Support-vector networks," Mach Learn, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[4] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. in Springer Series in Statistics. New York, NY: Springer New York, 2009. doi: 10.1007/978-0-387-84858-7.

[5] O. Saberi, "Fine-Tuning Machine Learning Regressors for Optimal Hyperparameter Selection," IAI600Lab3, GitHub, 2024. [Online]. Available: https://github.com/omisa69/IAI600Lab3. [Accessed: Oct. 2, 2024].