ASEAN
Journal of Scientific and Technological Reports
Online ISSN:2773-8752

Research article

Enhancing Linear Regression Through Neighbor-based Similarity Analysis

Chinnawat Chetcharungkit[1*]

[1] Department of Mathematics, Faculty of Science, Kasetsart University, Bangkok, 10903, Thailand
[*] Correspondence: fscicwc@ku.ac.th

Citation:
Chetcharungkit, C. Enhancing linear regression through Neighbor-based similarity analysis. *ASEAN J. Sci. Tech. Report.* **2024**, 27(5), e251974. https://doi.org/10.55164/ajstr.v27i5.251974

Article history:
Received: December 6, 2023
Revised: August 8, 2024
Accepted: July 30, 2024
Available online: September 7, 2024

Abstract: When working with real-world datasets characterized by complex and non-linear relationships, the limitations of non-complex machine learning models like linear regression become evident. In response to addressing this technical problem, we propose a novel algorithm to enhance linear regression without the necessity of complex mathematical or statistical expressions. Instead, the algorithm segments data into multiple subgroups or neighbors, each with its best-fitting line. The primary objective of this approach is to enable more accurate predictions for unseen data points by utilizing the most similar neighbors and their corresponding linear regression lines, with the support of k-nearest neighbors. Empirical evidence from three publicly available housing price datasets demonstrates the algorithm's effectiveness in improving traditional linear regression models.

Keywords: Linear modeling; machine learning; K-nearest neighbor; boosting algorithm

1. Introduction

In today's era of precise and rapid advancements in computational capabilities, various advanced machine learning techniques, like modified gradient boosting trees and deep learning, have emerged to improve model performance significantly [1, 2]. These sophisticated methods can model complex non-linear relationships in data and accommodate various data types such as images and text [3, 4]. With growing research, these techniques have led to predictive modeling across finance, healthcare, and marketing domains. Nonetheless, amidst the ongoing global warming crisis, it is recommended that complex models like deep learning be used only for essential purposes as they consume a lot of energy. This results in significant carbon dioxide emissions. For instance, it is found that training the bidirectional encoder representations from the transformers language model (BERT LM), an advanced deep learning model, can emit as much carbon dioxide as a year's home energy consumption [5]. This suggests that complex models should be used only when necessary.

In contrast to complex models, linear regression is a simple machine learning method that models the relationship between dependent variables and one or more independent variables by fitting a linear equation to observed data. While linear regression may not effectively handle data with non-linear relationships like deep learning, its role in data science persists. It remains a fundamental and indispensable method in statistics, data analysis, and financial engineering [6]. Its enduring importance lies in simplicity and interpretability. Since linear regression offers rapid implementation, requiring minimal computational resources and enabling swift model development compared to complex architectures of deep learning models, employing linear regression could extend global efforts to mitigate the threat of the global warming crisis. Also, due to its simplicity, individuals can easily understand and use linear regression, even with limited knowledge of machine learning. Moreover, linear regression demonstrates superiority when the relationships between variables are primarily linear. This is an example of a scenario where deep learning models cannot rival its performance. Yet, in data analysis, it becomes evident that many real-world datasets exhibit complex and non-linear relationships. For example, stock price data over time do not follow a straight line [7]. When we face such intricate data, it becomes essential to acknowledge the limitations of linear regression, as it may not be the most suitable approach. Nonetheless, rather than discarding linear regression, an attempt to enhance its capabilities to deal with data exhibiting non-linear patterns is intriguing. The success of this approach could eliminate the need for employing deep learning unless it is genuinely essential.

To date, numerous studies have sought to enhance regression models. The first approach belongs to a class of feature engineering intending to make the data more suitable and informative. This enables the model to learn patterns and make more precise predictions. The works in [8-10] are examples of this approach. However, feature engineering demands domain knowledge and a significant amount of time since we must carefully analyze the relationships between input features and target variables. This task can become challenging, especially when dealing with large input feature sets. The second approach is the so-called regularization. This technique adds a penalty term to the model's loss function [11-14]. The last approach introduces non-linearity to linear regression, such as using basis functions [15] and certain classes of statistical models like generalized linear models (GLMs) [16]. Unfortunately, the drawbacks of the latter two are that a solid knowledge of mathematics and statistics and coding skills are highly required. This complexity is further compounded when using standard platforms like scikit-learn in Python [17], as modifying the library's code as suggested by the mentioned papers might be nontrivial. Of course, these techniques may not suit every artificial intelligence (AI) developer, especially beginners.

Therefore, this research aims to develop a novel, user-friendly algorithm that can enhance the performance of linear regression models without involving the intricacies of daunting mathematical expressions, which might pose challenges to certain AI developers. Instead, we achieve this by incorporating a simple model like K-nearest neighbor (KNN) into our proposed algorithm. The algorithm's implementation is also designed to be straightforward, ensuring accessibility even for individuals not well-versed in coding. Three housing price datasets in [18-20] have been selected to assess the proposed algorithm's effectiveness. The choice of housing price data is due to its significance in various aspects. The first obvious reason is that they are standard and popular datasets for regression tasks. Apart from the technical reason, housing prices substantially impact the economy, as the real estate market plays a vital role in driving economic growth and stability [21]. In terms of modeling, some studies in [22-26] have attempted to construct machine learning models for housing price prediction. However, these studies relied on various common machine learning tools such as exploratory data analysis, linear regression, artificial neural networks, support vector machines, and gradient boosting trees, and none of them introduced novel methodologies.

2. Background

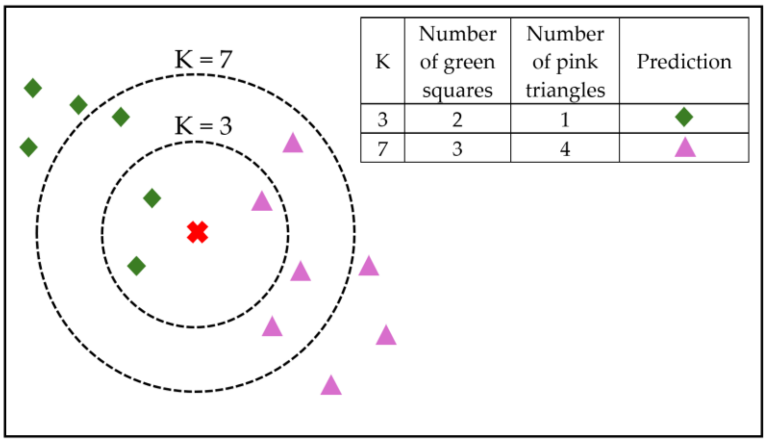
We commence by offering a concise overview of K-Nearest Neighbors (KNN), linear regression, and principal component analysis (PCA), as these methodologies will be subsequently applied in this research.

2.1 K-Nearest Neighbors (KNN)

K-nearest neighbors (KNN) is a straightforward and instinctive machine learning technique in classification and regression tasks. It is categorized within the class of instance-based, non-parametric approaches. The central idea behind KNN is that data points with similar features tend to belong to the same class or exhibit similar behaviors [15]. To determine similarity, in this study, we use the so-called Euclidean norm or L^2-norm defined as follows:

$$\|\boldsymbol{x}-\boldsymbol{y}\|_2 = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1}$$

where $\boldsymbol{x} = (x_1, \dots, x_n)$, $\boldsymbol{y} = (y_1, \dots, y_n) \in \mathbb{R}^n$. Intuitively, the closer two vectors are, the smaller their L^2-norm is. Hence, the predictive outcome for classification tasks is the class label, determined by selecting the most frequently occurring class among the K nearest neighbors. To clearly understand how KNN works, Figure 1 demonstrates how the K-Nearest Neighbors (KNN) algorithm classifies a new data point (the red cross). If K is set to be 3, the three nearest neighbors (inside the smaller dashed circle) include 2 green squares and 1 pink triangle, resulting in a prediction of a green square due to the majority. However, if K = 7, the seven nearest neighbors (inside the larger dashed circle) include 3 green squares and 4 pink triangles, leading to a prediction of a pink triangle instead. It can be noted that the value of the hyperparameter K plays a crucial role in determining the classification outcome by considering different sets of nearest neighbors. To find an optimal value of K, we can test various K values and choose the one that provides the best performance on a test set.



K	Number of green squares	Number of pink triangles	Prediction
3	2	1	◆
7	3	4	▲

Figure 1. The illustration of how the K-Nearest Neighbors (KNN) algorithm works to classify new data point, represented by the red cross, with K = 3 and 7.

2.2 Linear Regression

Let $\boldsymbol{X}$ be an input matrix in $\mathbb{R}^{n\times d}$ and $\boldsymbol{y}$ a target vector in $\mathbb{R}^n$. Linear regression aims to find a vector of parameters $\boldsymbol{w}$ in $\mathbb{R}^d$ such that $\|\boldsymbol{Xw} - \boldsymbol{y}\|_2^2$ is minimized as much as possible. It is well-known that the solution to this optimization problem is $\boldsymbol{w} = (\boldsymbol{X^T X})^{-1}\boldsymbol{X^T y}$ [27]. Note that since $\boldsymbol{w}$ is a result of matrix multiplication, it becomes clear that linear regression might exhibit poor performance in cases where the underlying pattern is not linear.

2.3 Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique used in data analysis and machine learning to transform a dataset into a new coordinate system, where the data's variance is maximized along the new axes. For example, if the original data has 200 input features, PCA can transform the dataset to reduce the number of features to any desired amount, such as 50. Moreover, with PCA, we can ensure that the new dataset with 50 input features effectively represents the original dataset.

PCA identifies the principal components (linear combinations of the original features) that capture the most significant variation in the data. These principal components are ordered by importance, allowing for the retention of the most informative components while reducing the dimensionality of the dataset. The algorithm for dimensionality reduction using PCA is as follows [15]:

Input Variables:

- **X**: Original standardized data matrix with dimensions $m \times n$ (where m is the number of samples and n is the number of features).
- k: Desired number of principal components, i.e., the number of features of the newly transformed data.

PCA Algorithm:

1. Calculate the covariance matrix ($\mathbf{\Sigma}$) of $\mathbf{X}$.

$$\mathbf{\Sigma} := \frac{1}{m}\mathbf{X}^{\mathrm{T}}\mathbf{X}$$

Decompose $\mathbf{\Sigma}$ using eigen-decomposition as a product of an orthogonal matrix $\mathbf{V}$ whose columns are the real-orthonormal eigenvectors of $\mathbf{\Sigma}$, and a diagonal matrix $\mathbf{\Lambda}$ whose entries are all the eigenvalues of $\mathbf{\Sigma}$.

$$\mathbf{\Sigma} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathrm{T}}$$

2. Sort the eigenvalues in $\mathbf{\Lambda}$ in descending order and rearrange the eigenvectors in $\mathbf{V}$ accordingly (if necessary).
3. Choose the top k eigenvectors corresponding to the largest k eigenvalues to form the matrix of principal components.

$$\mathbf{V}_k = \mathbf{V}[:, 1:k]$$

where $\mathbf{V}[:, 1:k]$ refers to the selection of the first k columns from the matrix $\mathbf{V}$.

4. The output (transformed) data is given by $\mathbf{X}\mathbf{V}_k$, which is a matrix of dimension $m \times k$.

PCA is generally admitted as a valuable tool for simplifying complex data, visualizing patterns, and removing multicollinearity in feature sets, making it a fundamental technique in data preprocessing and exploratory data analysis. In this study, PCA is an optional tool that is not integrated into any part of the proposed algorithm. Instead, we only employ it as a tool for data preprocessing to reduce the dimensionality of data to avoid prolonged computational processing.

3. Motivation and Methods

Before presenting the pseudocode algorithm of our proposed method, let us first delve into the idea behind it.

3.1 Motivation

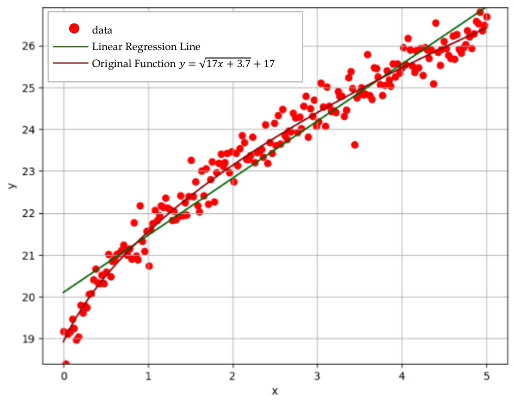


Figure 2. Scatter plot of 200 data points (red) with errors deviating from the equation $y = \sqrt{17x + 3.7} + 17$ (dark red) and the regression line (green).

In Figure 2, we simulate a scenario where the true model represents the relationship between the x-axis and y-axis through the equation $y = \sqrt{17x + 3.7} + 17$. After that, the 200 data points generated from the equation but with added noise or errors represent the data we collected. We also plot the linear regression line (in green) fitted from this data. Notably, the linear regression model did not perform well in this case, clearly due to the underlying non-linear pattern presented in the data.

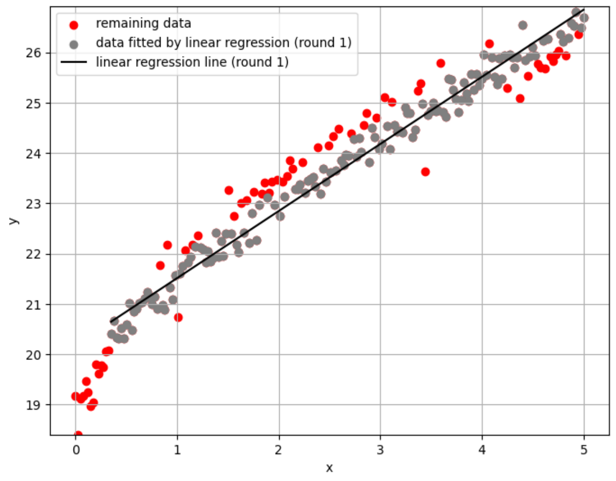


Figure 3. Round-1 linear regression line (black) fitted from the filtered data points (grey dots).

However, from Figure 3, if we pick only data points where the error between the predicted value and the actual value is less than the 70th percentile of the error observed in the entire dataset (in grey), the very linear regression line called round-1 linear regression line (in black), perform better with this filtered data.

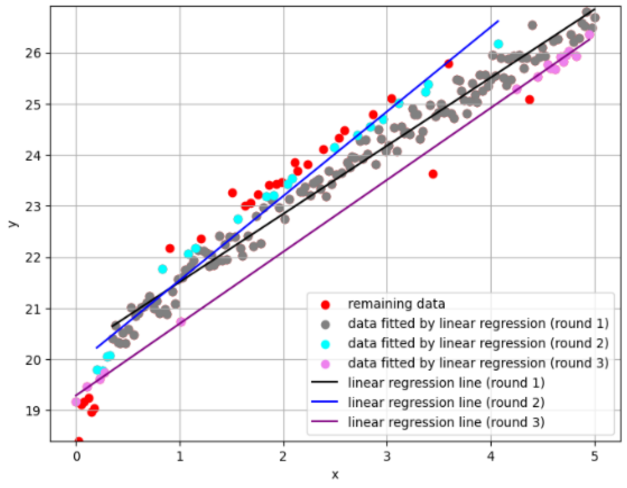


Figure 4. Round-2 and round-3 linear regression lines fitted from the corresponding filtered data points.

Next, we iterate the above step and yield the round-2 linear regression line (depicted in blue in Figure 4). It is obtained by fitting the remaining data in Figure 3, and subsequently, we select only the best-fitted data points (shown in cyan), as illustrated in Figure 4. We then repeat this process for one more round, yielding the round-3 linear regression line (in purple) and its corresponding data (in violet). Of course, this process will continue until all the data has been accounted for. Every data point has been assigned to an appropriate subgroup or neighbor and its corresponding linear regression line. At the end of this process, all the data is categorized into multiple neighbors, each with its own well-predicted linear regression line. When predicting the value of an unseen data point, we first determine which neighbor the new data point best fits through the k-nearest neighbor (KNN) model. Then, we employ that neighbor's corresponding linear regression line to receive the prediction. For example, suppose that the KNN model identifies a new data point resembling the cyan neighbor's pattern. Then, we utilize the blue linear regression line to forecast the expected outcome of this unseen data. The above approach would make more accurate predictions based on the characteristics of similar data points within the same neighborhood. These concepts form the foundation of the algorithm proposed by this research, as shown in Table 1.

Table 1. The proposed algorithm

Pseudocode

Input

iter	the number of iterations to obtain multiple neighbors of data and their corresponding linear regression lines
n_perc	the *n*th percentile in the dataset
num_knn	the number of nearest neighbors to use for k-nearest neighbors (KNN) model
X_train	training data
y_train	target variable of the training data
X_test	test data

Output

$X_1,\ldots,X_{iter}$	a sequence of neighbors of the data in each iteration
$l\_reg_1,\ldots,l\_reg_{iter}$	a sequence of the linear regression models corresponding to the neighbors of the data
y_pred	the predicted values from *X_test* by the proposed algorithm

Step 1: Find all the neighbors and build their corresponding linear regression models.

```
X = X_train
y = y_train
for i = 1 to iter :
    l_reg.fit(X, y)
    pred = l_reg.predict(X)
    dist = |pred - y|
    threshold_dist = n_perc in dist data
    if $i \neq iter$ :
        $X_i$ = X whose dist $\leq$ threshold_dist
        $l\_reg_i$ = l_reg
        X = X not containing $X_i$
        y = y which corresponds to X
    else:
        $X_i$ = X
        $l\_reg_i$ = l_reg
```

Step 2: Predict the target values.

```
# Determine the neighbor to which each data point in X_train belongs.
y_knn = [ i for x in X_train if x belongs to $X_i$ ])

# Build a KNN model
KNN.fit(X_train, y_knn, n_neighbors = num_knn)

# Forecast the outcome values
for $x_j$ in X_test :
    neighbor = KNN.predict($x_j$)    # To get the neighbor to which $x_j$ belongs
    $y\_pred_j$ = $l\_reg_{neighbor}$.predict($x_j$)    # Use the corresponding linear regression of neighbor to forecast
```

Table 1 shows three parameters for the algorithm: *iter*, *n_perc*, and *num_knn*. To find the optimal value for each parameter, we can experiment with various combinations of their settings to find the most effective one. This process is commonly referred to as "grid search." Iter aims to set the number of iterations required to obtain multiple neighbors of the data and their corresponding linear regression lines. N*_perc* removes data points that poorly fit the linear regression line. As for the last parameter, *num_knn* acts as a hyperparameter for the KNN model. Recall that KNN allows us to identify the most suitable neighbor to which any unseen data point belongs and then determine the linear regression line for prediction.

While KNN is integrated, this does not introduce additional complexity to the proposed algorithm. It can be noted that each line of code presented in Table 1 is just a routine command typically employed by data scientists. Furthermore, no modifications to the existing library code are required. All of these are to ensure that the implementation remains trouble-free for users.

Before applying the proposed algorithm to the datasets, it is vital to mention that Figure 4 serves only the purpose of visualizing how the algorithm works, even though concerns about overfitting might exist. However, this concern may not be as significant when dealing with real-world data because the simulated data in Figure 4 has only a quadratic pattern, less complex than any actual data. Also, linear regression is not a complex model. In particular, the overfitting issue is problematic because the model cannot accurately predict unseen data. Hence, if our algorithm can produce a model with better performance than the baseline model, it is legitimate to say the proposed algorithm is successful.

3.2 Methods

Now, we will employ the proposed algorithm to the three selected datasets. The process involves the following steps. For simplicity in evaluating the effectiveness of the proposed algorithm, some conventional steps like exploratory data analysis, data imputation to fill missing values, feature selection, and feature engineering will be omitted.

3.2.1 Data Preparation

The three datasets: house prices [18], California housing prices [19], and Boston house prices [20], underwent conventional techniques like dropping features with many missing values and removing rows containing missing values. All the categorical features were transformed using one-hot encoding (if necessary). Furthermore, in the case of the first dataset, the PCA algorithm in section 2.3 was applied due to its large number of input features (297) resulting from one-hot encoding. Setting the number of principal components to 150, the PCA-transformed data will have only 150 input features. Table 2 presents an overview of the preprocessing steps performed on each dataset during this phase.

Table 2. Summary of data characteristics and data preprocessing steps of the three datasets.

Item	House Prices	California Housing Prices	Boston House Prices
Size of raw data	1,094×76	20,433×10	506×14
Dropping column	Alley, PoolQC, MiscFeature, FireplaceQu, and Fence	None	None
Number of categorical features	42	1	0
PCA	Yes (150 principal components)	No	No
Number of input features	150	13	13
Target feature	SalePrice	median_house_value	MEDV

3.2.2 Train-test Split

Each dataset is split into training and test sets with a ratio of 67:33. The target feature for each dataset is shown in the final row of Table 2. This means we have *X_train, X_test, y_train,* and *y_test* for each housing dataset.

3.2.3 Model Training and Prediction

The proposed algorithm fits the input features (*X_train*) and the output feature (*y_train*) of the training dataset for each of the three datasets. However, since the proposed algorithm comprises three hyperparameters, *iter*, *n_perc*, and *num_knn*, different combinations of these values can yield various models with varying performance levels. Consequently, the optimal combination of these hyperparameter values is crucial. In this paper, to obtain the best parameter configuration for the algorithm, we loop through all the combinations of the following settings :

- *iter* ∈ {1, 2, 3 ,4, 5, 6, 7, 8},
- *n_perc* ∈ {0.3, 0.5, 0.7, 0.75},
- *num_knn* ∈ {1, 3, 17, 31, 59, 93}.

In more detail, we exhaustively tried all the 8×4×6 = 192 possible combinations from the given set of each hyperparameter to build 192 regression models and then predict the target values (*y_pred*) from the test datasets of the three housing datasets.

It is worth mentioning that the range or values of each hyperparameter can be set arbitrarily. The broader the ranges we specify, the higher the chance of finding more optimal parameters; however, this will require more time. This process involves a trade-off between the thoroughness of the search and the time required to perform it. While there is no guarantee of finding the absolute optimal solution, this approach ensures we identify the most optimal combination within the given set of hyperparameters.

3.2.4 Model Evaluation

Following the generation of 192 sets of predicted values (*y_pred*) of each housing dataset from the previous step, we compute several standard regression evaluation metrics using the *y_pred* values and the *y_test* values from subsection 3.2.2. These metrics will select the best model among the 192 models. The evaluation metrics include root mean square error (RMSE), R^2-score, mean absolute percentage error (MAPE), and mean absolute error (MAE), as follows:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i-\tilde{y_i})^2}{n}} \tag{2}$$

$$\mathrm{R}^2 = 1 - \frac{\sum_{i=1}^{n}(y_i-\tilde{y_i})^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2} \tag{3}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i-\tilde{y_i}}{y_i}\right| \times 100 \tag{4}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \tilde{y_i}| \tag{5}$$

where y_i is the *i*th actual output value (the *i*th value of *y_test*), $\tilde{y_i}$ is the *i*th predicted value (the *i*th value of *y_pred*), $\bar{y}$ is the average of the actual output values, and n is the number of rows of the test set. Note that the four metrics share the common goal of measuring how well a model's predictions align with the actual observed values, although their formulae are different, and the best model is the one with the highest R^2-score and the lowest RMSE, MAPE, and MAE.

4. Results and Discussion

The most optimal parameter configurations for *iter*, *n_perc*, and *num_knn* across the three selected housing price datasets, along with the corresponding values of the four evaluation metrics for regression, are shown in Table 3. The evaluation metrics obtained from the traditional (baseline) linear regression models to assess the effectiveness of the proposed algorithm are also provided in this table.

Table 3. Best parameter configurations and the four evaluation metrics across the three datasets.

Item		House Prices	California Housing Price	Boston House Prices
Best of (*iter*, *n_perc*, *num_knn*)		(3, 0.75, 17)	(5, 0.3, 59)	(6, 0.5, 1)
RMSE	Baseline	51,186.151	66,572.595	4.907
	Proposed	47,448.611	65,484.518	4.260
R^2	Baseline	0.535	0.665	0.747
	Proposed	0.601	0.676	0.809
MAPE	Baseline	0.155	0.280	0.189
	Proposed	0.139	0.274	0.162
MAE	Baseline	24,534.293	48,767.613	3.671
	Proposed	22,408.018	47,832.910	3.094

Table 4. Percentage change in the evaluation metric values for the proposed models compared to the baseline models across the different datasets.

Dataset	Metric	Percentage Change (%)
House Prices	RMSE	-7.302
	R^2	12.336
	MAPE	-10.323
	MAE	-8.667
California Housing Prices	RMSE	-1.634
	R^2	1.654
	MAPE	-2.143
	MAE	-1.917
Boston House Prices	RMSE	-13.185
	R^2	8.300
	MAPE	-14.286
	MAE	-15.718

Among the three datasets, the Boston housing dataset shows the most robust linear pattern, with the highest R^2-score from the baseline model. In contrast, the House prices dataset displays the weakest linear pattern. To see a clear comparison, we present Table 4, visually represented by Figure 5. The table provides the percentage changes in the four performance metrics for the proposed models compared to the baseline models across the three housing price datasets. Note that when it comes to RMSE, MAPE, and MAE, negative percentages show enhancements in performance, whereas positive ones indicate deteriorations. Yet, this situation becomes otherwise for R^2. This means that the proposed algorithm empirically enhances the performance of the traditional models.

Specifically, according to Figure 5, it is evident that the proposed algorithm significantly enhances the performance of the traditional linear regression model, achieving the best results in the case of the Boston house prices dataset, while the improvement is the least pronounced in the California housing price dataset. Furthermore, regardless of the underlying data pattern, the proposed model can effectively augment or replace the traditional linear regression model. This is evidenced by the substantial percentage changes in the evaluation metrics for the house prices dataset, which rarely exhibits a linear underlying pattern.

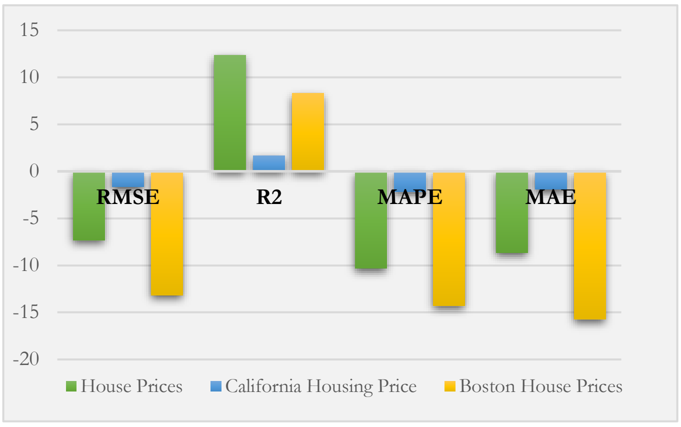


Figure 5. Percentage changes in evaluation metric values.

Examining the most challenging dataset for the proposed algorithm, the California housing price dataset, our initial observation reveals a small degree of linearity through an R^2-score of 0.665 from the baseline model. This first indicates a more complex underlying pattern on this dataset. Also, the optimal value of num_knn is extremely high, up to 59, reflecting that many data points are proximate. These issues lead to difficulty partitioning data into multiple neighbors and constructing best-fitting linear regression lines. Consequently, the proposed algorithm demonstrates its lowest effectiveness on this dataset. These findings show that this dataset would be better suited for advanced models like gradient-boosted trees or deep learning, even though the proposed algorithm can be deployed to enhance performance with modest improvements. From the results, it is evident that the proposed algorithm can significantly enhance linear regression. This improvement is particularly beneficial for fields that highly rely on linear regression, such as the Capital Asset Pricing Model (CAPM) in finance [6], or those who value linear regression's simplicity and interpretability. Yet, in some practical applications where precision is critical, it is advisable to consider other advanced regression models—such as support vector machines, gradient boosting trees, neural networks, and the proposed algorithm—to ensure that the most accurate and practical model is selected for real-world scenarios.

5. Conclusions

The study thoroughly evaluated the algorithmic approach to address the technical problem of linear regression's limitations when applied to non-linear data using three distinct housing price datasets: House Prices, California Housing Prices, and Boston House Prices. The results demonstrate that the proposed algorithm significantly improves the performance of the traditional linear regression model by decreasing RMSE, MAPE, and MAE metrics while increasing the R^2 score across all the datasets. The success of our proposed algorithm would be due to its capacity to break down complex datasets into smaller and manageable neighbors and then construct individual linear regression models for each of them. This approach allows us to capture finer patterns and relationships within the data, resulting in more accurate predictions with the support of KNN. The utility of applying the proposed algorithm improves the performance of linear regression models with a mathematically effortless approach. Still, it also contributes to the model selection process by indicating the suitability of inviting more complex models like deep learning in appropriate scenarios. Moreover, AI developers could gain advantages from the proposed algorithm by freely substituting their preferred model for linear regression. This might be another approach to enhance the performance of any existing reliable models.

6. Acknowledgements

This work is well supported by Department of Mathematics, Faculty of Science, Kasetsart University, for providing the facilities to conduct the research.

Author Contributions: Conceptualization, methodology, investigation, formal analysis, writing—original draft preparation, writing—review and editing, C.C.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] Moore, A.; Bell, M. XGBoost, A novel explainable AI technique, in the prediction of myocardial infarction: A UK Biobank cohort study. *Clinical Medicine Insights: Cardiology,* **2022**, *16*, 1-6. https://doi.org/10.1177/11795468221133611

[2] Khan, M. S.; Salsabil, N.; Alam, M. G. R., et al. CNN-XGBoost fusion-based affective state recognition using EEG spectrogram image analysis. *Scientific Reports* **2002**, *12*, 14122. https://doi.org/10.1038/s41598-022-18257-x

[3] Moraru, L.; Sistaninejhad, B.; Rasi, H.; Nayeri, P. A Review Paper about Deep Learning for Medical Image Analysis. *Computational and Mathematical Methods in Medicine*, **2023**, https://doi.org/10.1155/2023/7091301

[4] Shorten, C.; Khoshgoftaar, T. M.; Furht, B. Text Data Augmentation for Deep Learning. *Journal of Big Data*, **2021**, *8*, 101-135. https://doi.org/10.1186/s40537-021-00492-0

[5] Dodge, J.; Prewitt, T.; Combes, R. T. D., et al. Measuring the carbon intensity of AI in cloud instances. In 2022 ACM Conference on Fairness, Accountability, and Transparency, **2022**, 1877-1894. https://doi.org/10.1145/3531146.3533234

[6] Fama, E. F.; French, K. R. The Capital Asset Pricing Model: Theory and Evidence. *Journal of Economic Perspectives*, **2004**, *18*(3), 25-46. https://doi.org/10.1257/0895330042162430

[7] Singh, T.; Kalra, R.; Mishra, S., et al. An efficient real-time stock prediction exploiting incremental learning and deep learning. *Evolving Systems*, **2022**, *14*, 919-937. https://doi.org/10.1007/s12530-022-09481-x

[8] Walberg, H. J.; Rasher, S. P. Improving Regression Models. *Journal of Educational Statistics*, **1976**, *1*, 253-277. https://doi.org/10.2307/1164786

[9] Uddin, M. F.; Lee, J.; Rizvi, S., et al. Proposing Enhanced Feature Engineering and a Selection Model for Machine Learning Processes. *Applied Sciences*, **2018**, *8*(4). https://doi.org/10.3390/app8040646

[10] Qiao, Z.; Wang, C. H.; Liu, J. Y. Integrating Feature Engineering with Deep Learning to Conduct Diagnostic and Predictive Analytics for Turbofan Engines. *Mathematical Problems in Engineering*, **2022**. https://doi.org/10.1155/2022/9930176

[11] Deisenroth, M. P.; Faisal, A. A.; Ong, C. S. Mathematics for Machine Learning. Cambridge, UK: Cambridge University Press. **2020**.

[12] Arashi, M.; Roozbeh, M.; Hamzah, N. A., et al. Ridge regression and its applications in genetic studies. *PLoS ONE*, **2021**, 16. https://doi.org/10.1371/journal.pone.0245376

[13] Michel, V.; Gramfort, A.; Varoquaux, G., et al. Total Variation Regularization Enhances Regression-Based Brain Activity Prediction. *In First Workshop on Brain Decoding: Pattern Recognition Challenges in Neuroimaging*, **2021**, 9-12. https://doi.org/10.1109/WBD.2010.13

[14] Samavat, A.; Khalili, E.; Ayati, B., et al. Deep Learning Model with Adaptive Regularization for EEG-Based Emotion Recognition Using Temporal and Frequency Features. *IEEE Access*, **2022**, *10*, 24520-24527. https://doi.org/10.1109/ACCESS.2022.3155647

[15] Marsland, S. Machine Learning: An Algorithmic Perspective, 2nd ed. New York: Taylor & Francis Group, **2014**, 158-160.

[16] Dunn, P. K.; Smyth, G. K. Generalized linear models with examples in R. New York: Springer, **2018**. 211-233.

[17] Scikit-learn. (n.d.). Scikit-learn: Machine Learning in Python. Available: https://scikit-learn.org/stable/ [Accessed: 15 Jul 2023].

[18] Kaggle. (n.d.). House Prices: Advanced Regression Techniques. Available: https://www.kaggle.com/ competitions/house-prices-advanced-regression-techniques/data [Accessed: 1 Jul 2023].

[19] Kaggle. (n.d.). California Housing Prices. Available: https://www.kaggle.com/datasets/camnugent/ california-housing-prices [Accessed: 1 Jul 2023].

[20] Kaggle. (n.d.). Boston House Prices. Available: https://www.kaggle.com/datasets/vikrishnan/boston-house-prices [Accessed: 1 Jul 2023].

[21] Wang, H. Q.; Liang, L. Q. How Do Housing Prices Affect Residents' Health? New Evidence From China. Frontiers in Public Health, **2022**, *9*, 816372. https://doi.org/10.3389/fpubh.2021.816372

[22] Mukhlishin, M. F.; Saputra, R.; Wibowo, A., et al. Predicting house sale price using fuzzy logic, Artificial Neural Network and K-Nearest Neighbour. In Proceedings of the 2017 1st International Conference on Informatics and Computational Sciences, **2017**, 171-176. https://doi.org/10.1109/ICICOS.2017.8276357

[23] Phan, T. D. Housing Price Prediction Using Machine Learning Algorithms: The Case of Melbourne City, Australia. In Proceedings of the 2018 International Conference on Machine Learning and Data Engineering, **2018**, 35-42. https://doi.org/10.3390/land11112100

[24] Truong, Q.; Nguyen, M.; Dang, H., et al. Housing Price Prediction via Improved Machine Learning Techniques. *Procedia Computer Science*, **2020**, 433-442. https://doi.org/10.1016/j.procs.2020.06.111

[25] Gupta, P.; Zhang, Q. Housing Price Prediction Based on Multiple Linear Regression. Scientific Programming, **2021**. https://doi.org/10.1155/2021/7678931

[26] Tanamal, R.; Minoque, N.; Wiradinata, T., et al. House Price Prediction Model Using Random Forest in Surabaya City. *TEM Journal*, **2023**, *12*, 126-132. https://doi.org/10.18421/TEM121-17

[27] Goodfellow, I. J.; Bengio, Y.; Courville, A. Machine Learning Basics. In Deep Learning. Cambridge: MIT Press, **2016**, 96-146.