Residual Aware Stacking: A Novel Approach for Improved Machine Learning Model Performance

Hardev Ranglani[1]

Traditional stacking ensembles in Machine Learning aggregate predictions from multiple models to improve accuracy, but they often fail to address the residual errors left by base models. This paper introduces Residual-Aware Stacking (RAS), a novel approach that trains additional models to predict residuals (errors) of base models, creating a second layer of predictions. These, combined with the original base model predictions, are used to train a meta-model for the final output. This meta-model leverages both the original predictions and residual corrections to produce the final output. We demonstrate the improved accuracy and robustness of this technique by applying it on various regression datasets and comparing their performance against traditional models. This method highlights the potential of residual modeling in enhancing ensemble learning.

Keywords: Stacking, Ensemble Methods, Residual Models, Meta Learners, Regression Modeling

1 Introduction

Ensemble methods in Machine Learning use multiple algorithms to obtain better predictive performance than could be obtained from a single algorithms alone. They offer robust solutions to complex predictive tasks by combining the strengths of multiple models. Among these methods, stacking ensembles stand out for its ability to aggregate predictions from diverse base learners through a meta-model, potentially capturing intricate relationships between the base predictions and the target variable. This flexibility has made stacking a powerful approach for tasks in regression and classification alike [Wolpert, 1992]. However, traditional stacking methods often overlook residual errors—differences between actual and predicted values—that persist after the initial predictions. Modeling these residuals could uncover patterns missed by the base models, enhancing the ensemble's overall predictive performance.

This paper introduces Residual-Aware Stacking (RAS), a novel idea to this stacking ensemble framework that integrates residual learning. In this framework, the first layer consists of base models trained on the original dataset to generate initial predictions.

David C. Wyld et al. (Eds): BDAB, SVC, AIBD, ACSTY, NATP – 2025
pp. 175-187, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2024.150115

These predictions are then used to calculate residuals, which then serve as the target for a second layer of models (residual models). By explicitly addressing these residuals, the second layer of models focuses on correcting the errors of the base models. The predictions from both the base models and the residual models are combined to form a rich meta-dataset, which is used to train a meta-model that produces the final predictions. These final predictions are tested on the validation dataset by creating the corresponding meta-dataset and evaluating the prediction accuracy.

This RAS approach builds on ideas from residual modeling and error correction techniques. Residual learning has been widely applied in neural networks, such as in ResNet architectures for image processing [He et al., 2016], where it has proven effective in mitigating vanishing gradients. Similarly, gradient boosting methods like XGBoost [Chen & Guestrin, 2016] rely on iterative learning through residuals of successive models to refine predictions. However, integrating model residuals into a stacking ensemble framework for regression tasks has not been thoroughly explored. This paper addresses this gap by combining the strengths of stacking ensembles and residual learning to create a unified, robust predictive model.

The RAS framework has many advantages over the traditional stacking technique. First, the RAS framework reduces bias and variance by focusing on the errors left unaddressed by base models, improving the ensemble's generalization to unseen data [Dietterich, 2000]. Second, the method enhances the interpretability of the ensemble by explicitly modeling both initial predictions and residual corrections. Third, it is adaptable to various regression problems, where relationships between features and target variables can possibly be highly nonlinear or complex.

To evaluate the effectiveness of this framework, we apply it to multiple regression datasets, including the Allstate Claims dataset[Kaggle Allstate Claims dataset, 2018], the Diamond Prices prediction dataset[Kaggle Diamonds Price Dataset, 2022] and the Flight Prices prediction dataset[Kaggle Flight Prices Dataset, 2021]. Comparative analysis against traditional stacking, bagging, and boosting methods demonstrates that RAS consistently achieves superior performance, underscoring its potential as a versatile and powerful tool for regression tasks.

The remainder of this paper is organized as follows. Section 2 reviews related work on stacking ensembles, residual modeling, and ensemble learning. Section 3 describes the Residual-Aware Stacking framework, including implementation details and theoretical underpinnings. Section 4 presents the experimental setup and results, followed by a discussion in Section 5. Finally, Section 6 concludes with insights and potential directions for future research.

2 Literature Review

In this section, we review the foundational concepts and advancements in stacking ensembles, residual modeling, and ensemble learning, providing a background for the proposed Residual-Aware Stacking (RAS) framework.

2.1 Stacking Ensembles

Stacking ensembles, introduced by Wolpert [1992], involve combining predictions from multiple base models using a meta-model to improve generalization. Unlike bagging [Breiman, 1996] and boosting [Schapire, 1990], which aim to reduce variance or bias, stacking focuses on optimizing the combination of diverse base models.

Recent advancements in stacking have explored various meta-model configurations. Sill et al. [2009] proposed blending, a variation of stacking that uses holdout datasets for meta-model training. Notable implementations in large-scale applications include Google's use of stacking in its winning solution for the Netflix Prize [Koren et al., 2009]. More recently, auto-stacking frameworks like Auto-sklearn [Feurer et al., 2015] and H2O.ai's AutoML [LeDell & Poirier, 2020] have automated the design of stacking ensembles for various tasks.

However, traditional stacking methods often fail to address residual errors explicitly, limiting their ability to refine predictions. This paper builds on these works by incorporating residual learning into the stacking process.

2.2 Residual Modeling

Residual modeling has played an important role in advancing machine learning, particularly in gradient boosting and neural network architectures. Gradient boosting algorithms, such as XGBoost [Chen & Guestrin, 2016], LightGBM [Ke et al., 2017], and CatBoost [Prokhorenkova et al., 2018], iteratively refine predictions by minimizing residual errors at each step. These methods have demonstrated remarkable success in structured data tasks, such as competition-winning Kaggle solutions [Kaggle, 2021].

Residual learning also plays an important role in neural networks. The ResNet architecture [He et al., 2016] introduced skip connections to enable residual learning in deep networks, addressing the vanishing gradient problem and enabling the training of very deep architectures. While residual learning has been extensively studied in boosting and neural networks, its integration with stacking ensembles remains underexplored, providing the motivation for this work.

2.3 Ensemble Learning

Ensemble learning methods combine multiple models to improve robustness, accuracy, and generalization [Dietterich, 2000]. Main methods include:

1. **Bagging:** Ensemble method that reduces variance by training models on bootstrap samples and aggregating predictions, as in random forests [Breiman, 2001].

2. **Boosting:** Ensemble method that reduces bias by sequentially training models to correct errors of previous models, as in AdaBoost [Freund & Schapire, 1997].

Meta-learning approaches have further enhanced ensemble methods. For instance, dynamic ensemble selection techniques like META-DES [Cruz et al., 2018] dynamically select base models based on local competence. Similarly, multi-layer ensembles like Cascade Forest [Zhou & Feng, 2017] use hierarchical structures to refine predictions.

Despite these advancements, ensemble methods rarely address the residual errors left after the combination of base model predictions. The proposed RAS framework addresses this gap by explicitly modeling residuals in a two-tier stacking process, offering a novel contribution to ensemble learning.

2.4 Applications of Stacking and Residual Learning

Stacking and residual learning have found widespread applications across domains. In time-series forecasting, stacking ensembles have been used to combine ARIMA models with machine learning methods [Zhang, 2003]. Residual learning has improved

image processing tasks, such as object detection [Ren et al., 2017]. In structured data, gradient boosting algorithms like XGBoost have been widely used for regression and classification tasks [Chen & Guestrin, 2016].

The integration of residual learning into stacking ensembles can unlock new possibilities for addressing complex regression tasks, particularly in domains with nonlinear relationships and high-dimensional data. This paper extends these ideas, demonstrating the practical benefits of residual-aware stacking on multiple regression benchmarks.

3 Methodology

This section presents the methodology for the proposed Residual-Aware Stacking (RAS) framework. The goal of RAS is to extend the traditional stacking ensemble by incorporating residual modeling into the learning process, addressing limitations in traditional methods that fail to explicitly handle errors left by base models. RAS combines base model predictions and their residual corrections to produce more accurate and robust predictions through a meta-model.

3.1 Novelty of the Residual-Aware Stacking Framework

Traditional stacking ensembles focus on combining predictions from multiple base models through a meta-model to improve overall predictive accuracy. While this method effectively aggregates diverse modeling approaches, it does not explicitly address the residual errors—the differences between the actual and predicted values—that persist after base model predictions. These residuals often contain meaningful patterns that could improve model performance if leveraged correctly.

The **novelty of RAS** lies in its integration of residual learning into the stacking framework. RAS introduces an additional layer of residual models, which are specifically trained to predict the residuals of each base model. By explicitly modeling these residuals, RAS creates a richer set of features for the meta-model, allowing it to make more informed decisions when combining predictions. This dual-layer approach—combining the original predictions with residual corrections—is not commonly explored in traditional stacking, making RAS a significant extension.

3.2 Overview of the Residual-Aware Stacking Framework

The RAS framework operates in three primary stages:

1. Base Model Training and Initial Predictions: A diverse set of base ML models are used to predict the target variable. These preditcions are then used to compute the residuals for both training and validation data.

2. Residual Model Training: The difference between the actual values and the predictions from each base model are calculated as residuals. An additional set of models (referred to as residual models) are then trained to predict these residuals.

3. Meta-Model Training and Final Prediction: Predictions from the base models and residual models are combined to create a meta-dataset. This meta-dataset has the predictions from each of the base ML models, along with the residuals for these predictions, as predicted by the residual models as inputs. The target variable

for prediction here is the original target variable from the dataset Meta-models are then trained on this meta-dataset to produce the final predictions. A similar meta-dataset is created for the test data as well to evaluate the performance of the meta model.

3.3 Base Model Training

The first step involves training a diverse set of base models on the training dataset to predict the target variable. Each model independently learns a mapping from the input features X to the target variable y. Let $\hat{y}_i^{(m)}$ denote the predictions of the $m-th$ base model on data point i. The predictions from these models serve as the first layer of the ensemble.

We utilize several base models to capture different aspects of the data:

1. **Linear Regression (LR)**: Captures linear relationships between features and the target.
2. **Decision Tree (DT)**: Captures nonlinear relationships using hierarchical splits.
3. **K-Nearest Neighbors (KNN):** Learns patterns based on proximity in feature space.
4. **Random Forest (RF)**: An ensemble of decision trees trained on bootstrap samples.
5. **Gradient Boosting (GB):** Sequentially builds models to minimize residual errors.
6. **Support Vector Machines (SVM):** Learns optimal hyperplanes for regression.
7. **XGBoost (XGB):** An efficient implementation of gradient boosting.

For each base model M_m, we calculate predictions on the training data:

$$\hat{y}^{(m)} = M_m(X_{train})$$

Each base model generates predictions for both the training and validation datasets. These predictions represent the base model's estimate of the target variable. However, no model is perfect, and errors—known as residuals—remain. The residuals are calculated as the difference between the actual target values and the predictions from each base model. These residuals are critical, as they indicate patterns in the data that the base models failed to capture.

3.4 Residual Model Training

In the second stage, additional models are trained to predict these residuals. The purpose of these residual models is to address the errors left by the base models. By training on the residuals, these models focus on the nuances and patterns missed in the first layer.

For each base model, residual models, using the Random Forest algorithm, are trained using the residuals as the target variable. For example, if a decision tree model predicts house prices but systematically underestimates prices for larger homes, a residual model trained on the decision tree's residuals can help correct this specific

error. This process is repeated for all base models, ensuring that the second layer of the ensemble focuses on error correction rather than redundant learning. These residual predictions are then added as new features to the ensemble.

Once the base models are trained, we compute their residuals:

$$r_i^{(m)} = y_i - \hat{y}_i^{(m)}$$

These residuals represent the errors left unaddressed by each base model. The residuals serve as the target variable for training residual models.

We train additional models to predict the residuals of each base model, using the Random Forest algorithm for all residual models. Let $\hat{r}_i^{(m)}$ denote the predictions of the residual model for the m-th base model. The residual model thus learns:

$$\hat{r}_i^{(m)} = M_m(X_{val})$$

3.5 Meta-Model Training

In the final stage, predictions from both the base models and residual models are combined into a meta-dataset. This meta-dataset contains as input the original predictions from the base models as well as the predictions from the residual models, which are designed to correct the errors of the base models. The original target variable from the dataset is added to this meta-dataset as the target variable to be predicted. This meta-dataset is created both for the train data(to train the meta-learner) and the test data(to evaluate the performance of the meta-learner)

A meta-model is then trained on this enriched meta-dataset. The meta-model learns to optimally combine the original predictions and the residual corrections to produce the final predictions. For instance, if a residual model corrects a specific bias in one base model, the meta-model can learn to weigh this correction more heavily for certain types of data.

For the meta-model, we explore various approaches such as linear regression for simplicity and interpretability, random forest for capturing complex relationships, and decision trees for their ability to handle hierarchical decision-making. The meta-model aggregates the insights from both layers of the ensemble to produce the final output.

The meta-dataset contains:

1. Predictions from the base models: $\hat{y}_i^{(m)}$ for $m = 1, 2, 3, ...M$
2. Predictions from the residual models: $\hat{r}_i^{(m)}$ for $m = 1, 2, 3, ...M$

The meta-dataset is used to train a meta-model M_{meta}, which learns to combine these features into a final prediction:

$$\hat{y}_i^{(m)} = M_{meta}([\hat{y}_i^{(1)}, \hat{y}_i^{(2)}, ..., \hat{y}_i^{(M)}, \hat{r}_i^{(1)}, \hat{r}_i^{(2)}, ..., \hat{r}_i^{(M)}])$$

3.6 Prediction on Test Data

Once the RAS framework is trained, predictions for the test data are generated in three steps:

1. The base models provide initial predictions for the test data.

2. The residual models generate corrections by predicting residuals for each base model.
3. These predictions are combined into a meta-dataset, which the trained meta-model uses to generate the final predictions.
4. These predictions are compared to the original predictions from the base models using various metrics to evaluate the performance of these models

Thus, the Residual-Aware Stacking framework offers the following key advantages:

1. Error Correction: By explicitly modeling residuals, RAS addresses errors that traditional stacking methods may overlook.
2. Bias and Variance Reduction: Combining base and residual predictions reduces both bias and variance, leading to better generalization on unseen data.
3. Flexibility: The framework can incorporate diverse base and residual models, allowing it to adapt to various types of data and prediction tasks.
4. Interpretability: The layered structure provides clear insights into how base models and residual corrections contribute to the final prediction.

3.7 Evaluation Metrics

To comprehensively evaluate and compare the performance of the proposed Residual-Aware Stacking (RAS) framework with traditional machine learning models, we employ a range of metrics that capture various aspects of prediction accuracy and robustness. These metrics include absolute errors, squared errors, and percentage-based errors, providing a holistic view of model performance across different datasets and error distributions.

1. **Median Absolute Error** and **Mean Absolute Error** are used to measure the average absolute differences between predicted and actual values. MedAE is robust to outliers, while MAE provides an overall view of the typical error magnitude.

$$\text{MedAE} = \text{median}\left(|y_i - \hat{y}_i|\right)$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

2. **Median Squared Error (MedSE)** and **Mean Squared Error (MSE)** are used to emphasize larger errors by squaring the residuals. MedSE offers a robust measure, while MSE is more sensitive to outliers.
3. Square Root versions of these metrics, **Root Median Squared Error (RMdSE)** and **Root Mean Squared Error (RMSE)**, provide error magnitudes in the same units as the target variable, making them easier to interpret.

$$\text{MedSE} = \text{median}\left((y_i - \hat{y}_i)^2\right)$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$\text{RMSE} = \sqrt{\text{MSE}}$$

and

$$\text{RMdSE} = \sqrt{\text{MedSE}}$$

4. **Median Absolute Percentage Error (MedAPE)** and **Mean Absolute Percentage Error (MAPE)** are used to measure errors relative to the magnitude of the target values, expressed as percentages. MedAPE is robust to extreme percentage errors, while MAPE provides a general indication of prediction accuracy proportional to the target values.

$$\text{MedAPE} = \text{median}\left(\frac{|y_i - \hat{y}_i|}{y_i} \times 100\right)$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|y_i - \hat{y}_i|}{y_i} \times 100\right)$$

Overall, the RAS framework enhances traditional stacking by incorporating residual learning into the ensemble process. By combining predictions from base models and residual models in a meta-model, RAS effectively reduces prediction errors and improves robustness, making it a versatile tool for complex regression tasks.

4 Results

The Residual-Aware Stacking (RAS) framework was applied to three diverse regression datasets: the Allstate Claims Loss dataset, the Flight Prices Prediction dataset, and the Diamond Prices dataset. The data was divided into a 75-25 train-test split and the base models were used to get the initial model predictions. These predictions were then subtracted from the target variable to get each base model's residuals to train the residual models. Meta-models were then trained using these predictions and residuals as input to predict the target variable. The same meta-dataset was created on the test data to get the final predictions for the test data for the RAS framework. The goal was to evaluate the effectiveness of RAS in improving predictive accuracy compared to individual machine learning models and traditional stacking ensembles on the test dataset. The performance metrics considered were Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and Median Absolute Percentage Error (MedAPE), with an emphasis on demonstrating the incremental improvements achieved by the RAS meta-model.

4.1 Performance on the Allstate Claims Loss Dataset

The Allstate Claims Loss dataset, which involves predicting the severity of insurance claims, posed a challenge due to its high-dimensional and complex feature space. As shown in table 1 and table 2, the base machine learning models and the RAS framework meta models performed well, with the metrics calculated for each of the algorithm in table 1, with the bold metrics highlighting the model with the best performance for the given metric. The RAS meta models perform better on the metrics Median Absolute Error and Median Absolute Percentage Error, as these metrics are less sensitive to outliers, showcasing the superiority of the RAS meta models compared to base meta machine learning models.

	LinearRegression	DecisionTree	KNN	RandomForest	GradientBoosting	Lightgbm	XGBoost
MedAE	1,759.5	1,050.9	1,050.6	1,175.1	1,106.6	1,193.2	1,137.8
MAE	2,353.7	2,033.7	2,033.3	2,032.2	2,037.1	**2,029.1**	2,031.2
RMdSE	613.7	128.5	136.6	126.8	127.8	**126.5**	128.7
RMSE	1,487.9	1,174.8	1,179.8	**981.3**	1,046.9	991.2	997.5
MedSE	376,617.5	16,508.0	18,671.0	16,074.3	16,335.4	**16,002.8**	16,566.6
MSE	2,213,951.4	1,380,090.1	1,392,035.6	**963,041.1**	1,096,005.8	982,488.4	995,080.6
MedAPE	151.54%	90.52%	90.49%	101.21%	95.31%	102.77%	98.00%
MAPE	202.72%	175.16%	175.13%	175.03%	175.45%	**174.76%**	174.95%

Table 1: Performance comparison of various base machine learning models the Allstate Claims Loss Dataset

	LinearRegression_MetaModel	RF_MetaModel	DecisionTree_MetaModel
MedAE	1,085.8	1,020.2	**999.6**
MAE	2,039.6	2,043.8	2,043.8
RMedSE	129.2	139.2	143.0
RMSE	1,072.6	1,075.2	1,080.4
MedSE	16,686.1	19,386.3	20,435.6
MSE	1,150,401.5	1,156,040.3	1,167,371.0
MedAPE	93.52%	87.87%	**86.09%**
MAPE	175.67%	176.03%	176.03%

Table 2: Performance comparison of Residual Aware Stacking Meta models on the Allstate Claims Loss Dataset

4.2 Performance on the Flight Prices Prediction Dataset

For the Flight Prices Prediction dataset, which involves forecasting ticket prices based on factors like airline, duration, and stops, the models showed significant variation in performance. Here, the RAS meta models performed better in 5 out of the 8 metrics, as shown in table 4, thus showcasing enhanced performance compared to the base ML models, captured in table 3.

	LinearRegression	DecisionTree	KNN	RandomForest	GradientBoosting	Lightgbm	XGBoost
MedAE	2,637.5	147.3	357.0	170.1	95.4	410.7	110.6
MAE	3,973.9	793.3	1,573.0	770.2	**643.5**	1,112.4	683.4
RMedSE	2,637.5	147.3	357.0	170.1	95.4	410.7	110.6
RMSE	6,160.1	2,212.2	3,441.2	2,018.4	**1,988.6**	2,255.4	2,058.0
MedSE	6,956,284.6	21,698.6	127,449.0	28,919.2	9,094.0	168,694.2	12,231.0
MSE	37,946,852.9	4,893,962.2	11,841,893.9	4,073,756.8	**3,954,420.0**	5,087,005.9	4,235,235.0
MedAPE	23.20%	1.82%	4.44%	2.09%	1.13%	4.74%	1.29%
MAPE	41.41%	4.26%	8.08%	4.16%	3.30%	7.04%	3.49%

Table 3: Performance comparison of various base machine learning models the Flight Prices Prediction Dataset

4.3 Performance on the Diamond Prices Dataset

The Diamond Prices dataset presented a different set of challenges, involving nonlinear relationships between features like carat, cut, and clarity and the target variable, price. Here, the RAS meta models achieved better performance in 4 out of the 8 metrics as compared to the base ML models. All 4 of these metrics where the RAS meta models performed better are related to median as the central measure, indicating they are less sensitive to outliers.

	LinearRegression_Meta	RF_Meta	DecisionTree_Meta
MedAE	111.1	64.0	**61.0**
MAE	695.4	661.5	668.4
RMedSE	111.1	64.0	**61.0**
RMSE	2,097.9	2,089.5	2,108.5
MedSE	12,337.4	4,096.0	**3,721.0**
MSE	4,401,262.9	4,365,931.9	4,445,607.6
MedAPE	1.29%	0.84%	**0.83%**
MAPE	3.52%	**3.27%**	3.35%

Table 4: Performance comparison of Residual Aware Stacking Meta models on the Flight Prices Prediction Dataset

	LinearRegression	DecisionTree	KNN	RandomForest	GradientBoosting	Lightgbm	XGBoost
MedAE	568.4	89.8	150.3	79.4	79.5	122.9	78.7
MAE	780.2	262.7	463.3	**216.9**	233.4	246.4	225.1
RMedSE	568.4	89.8	150.3	79.4	79.5	122.9	78.7
RMSE	1,185.9	591.0	1,031.5	**459.3**	517.6	462.0	493.5
MedSE	323,127.7	8,069.7	22,590.1	6,304.4	6,327.2	15,101.0	6,199.7
MSE	1,406,377.0	349,286.4	1,063,904.5	**210,964.7**	267,938.6	213,423.7	243,497.9
MedAPE	19.58%	4.05%	7.01%	3.74%	3.57%	5.98%	3.58%
MAPE	43.88%	5.86%	11.45%	**4.90%**	5.06%	6.85%	4.99%

Table 5: Performance comparison of various base machine learning models the Diamond Prices Prediction Dataset

	LinearRegression_Meta	RF_Meta	DecisionTree_Meta
MedAE	**78.7**	80.5	81.0
MAE	227.7	226.2	229.1
RMedSE	**78.7**	80.5	81.0
RMSE	501.7	491.4	499.6
MedSE	**6,198.5**	6,484.4	6,561.0
MSE	251,733.3	241,497.5	249,581.1
MedAPE	**3.48%**	3.51%	3.54%
MAPE	4.96%	4.98%	5.05%

Table 6: Performance comparison of Residual Aware Stacking Meta models on the Diamond Prices Prediction Dataset

4.4 Discussion

Across all three datasets, the RAS framework outperformed individual machine learning models in many of the metrics. In all the cases, the RAS framework performed better when median is chosen as the measure of central tendency as opposed to the mean, indicating that the error metrics are sensitive to outliers. While the improvements over the best base models were marginal in absolute terms, they were highly consistent across all metrics and datasets. The results illustrate several key advantages of the RAS framework:

1. **Enhanced Error Correction:** By incorporating residual learning, RAS effectively addressed systematic errors left by base models, leading to noticeable reductions in MSE and RMSE.

2. **Improved Robustness:** The reduction in MedAPE across datasets highlights the ability of RAS to handle outliers and noisy data effectively.

3. **General Applicability:** The performance gains were observed in datasets with varying characteristics—high-dimensional insurance data, time-sensitive flight pricing data, and nonlinear diamond pricing data—demonstrating the versatility of RAS

These results validate the RAS framework as a practical and effective enhancement to traditional stacking, particularly for regression tasks where base models leave residual errors unaddressed. While the improvements over base models and traditional stacking were incremental, they were achieved without introducing significant computational complexity, making RAS a compelling choice for real-world applications.

5 Conclusion & Future Work

In this study, we introduced the Residual-Aware Stacking (RAS) framework, an enhancement to traditional stacking ensembles that integrates residual modeling to address errors left by base models. By explicitly modeling residuals, RAS effectively reduced prediction errors and improved robustness, as demonstrated on three diverse regression datasets: Allstate Claims Loss, Flight Prices Prediction, and Diamond Prices. The meta-model in RAS consistently outperformed individual models, showing superior accuracy across multiple evaluation metrics, even though the improvement in the metrics was marginal. These results highlight the potential of residual-aware stacking for tackling complex regression tasks.

The RAS framework can be adapted for classification problems, where residual modeling could focus on misclassification probabilities or error patterns, potentially improving performance in tasks like medical diagnosis or fraud detection. Extending RAS to real-time systems or streaming data scenarios is another promising avenue, allowing it to be used in fields such as financial forecasting or IoT applications. While this study focused on regression tasks, the principles of residual modeling within a stacking framework are broadly applicable and offer a foundation for further research. By exploring these future directions, RAS can become a versatile tool for addressing a wide range of predictive challenges in both regression and classification contexts.

6 References

1. Wolpert, D. H. (1992). Stacked generalization. Neural Networks, 5(2), 241-259.

2. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 770-778).

3. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD) (pp. 785-794).

4. Dietterich, T. G. (2000). Ensemble methods in machine learning. In International Workshop on Multiple Classifier Systems (pp. 1-15). Springer.

5. Kaggle (2016). Allstate Claims Severity dataset. Available at: https://www.kaggle.com/c/allstate-claims-severity

6. Kaggle Diamonds Price Dataset(AmirHossein Mirzaei) https://www.kaggle.com/datasets/amirhosseinmirzaie/diamonds-price-dataset/data

7. Kaggle Flight Price Prediction Dataset(Shubham Bathwal) https://www.kaggle.com/datasets/shubhambathwal/flight-price-prediction

8. Candanedo, L. M., Feldheim, V., & Deramaix, D. (2017). Data driven prediction models of energy use of appliances in a low-energy house. Energy and Buildings, 140, 81-97.

9. Harrison, D., & Rubinfeld, D. L. (1978). Hedonic prices and the demand for clean air. Journal of Environmental Economics and Management, 5(1), 81-102.

10. Breiman, L. (1996). Bagging predictors. Machine Learning, 24(2), 123-140.

11. Schapire, R. E. (1990). The strength of weak learnability. Machine Learning, 5(2), 197-227.

12. Sill, J., Takacs, G., Mackey, L., & Lin, D. (2009). Feature-weighted linear stacking. arXiv preprint arXiv:0911.0460.

13. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

14. Feurer, M., et al. (2015). Efficient and robust automated machine learning. Advances in Neural Information Processing Systems (NeurIPS).

15. LeDell, E., & Poirier, S. (2020). H2O AutoML: Scalable automatic machine learning. In ICML Workshop on AutoML.

16. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD).

17. Ke, G., et al. (2017). LightGBM: A highly efficient gradient boosting decision tree. In Advances in Neural Information Processing Systems (NeurIPS).

18. Prokhorenkova, L., et al. (2018). CatBoost: Unbiased boosting with categorical features. In Advances in Neural Information Processing Systems (NeurIPS).

19. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

20. Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1), 119-139.

21. Cruz, R. M. O., et al. (2018). META-DES: A dynamic ensemble selection framework using meta-learning. Pattern Recognition, 48(5), 1925-1935.

22. Zhou, Z. H., & Feng, J. (2017). Deep forest: Towards an alternative to deep neural networks. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI).

23. Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, 159-175.

24. Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137-1149.