Data Description and Exploratory Analysis

The Abalone dataset, sourced from the UC Irvine Machine Learning Repository, is used to predict an abalone's age, calculated as the number of rings plus 1.5. It consists of 4,177 observations with no missing values and includes eight predictor variables: one categorical (Sex) and seven numerical (Length, Diameter, Height, Whole Weight, Shucked Weight, Viscera Weight, and Shell Weight), measured in millimeters or grams. The target variable, Rings, represents the count of growth rings. This dataset is widely used for predictive modeling in regression and classification tasks.
The density plot of Rings shows an approximately bell-shaped distribution with a peak around 9–10 rings, indicating that most abalones in the dataset are 10.5 to 11.5 years old. The distribution has a slight right skew, as the mean is slightly higher than the median due to some older abalones. Violin plots of Sex categories show that male and female abalones have overlapping distributions, with medians around 10 rings and a wider spread, while infants have a lower median and less variation. The correlation plot reveals that most predictor variables are highly correlated with each other but not strongly correlated with Rings. Additionally, the distribution plots of Whole Weight, Shucked Weight, Viscera Weight, and Shell Weight show a right-skewed pattern, likely due to their high correlation. Even though the predictor variables are highly correlated, we will keep all of them for this analysis since each feature provides valuable information.

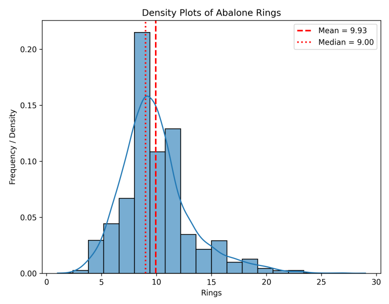

Figure 1: Distribution of Target Variable

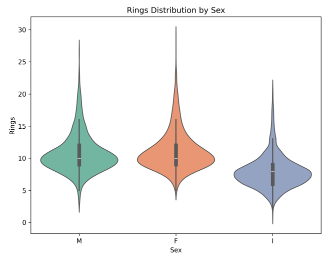

Figure 2: Categorical Variable

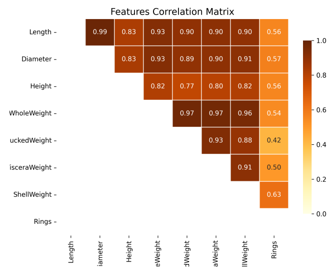

Figure 3: Correlation Heatmap

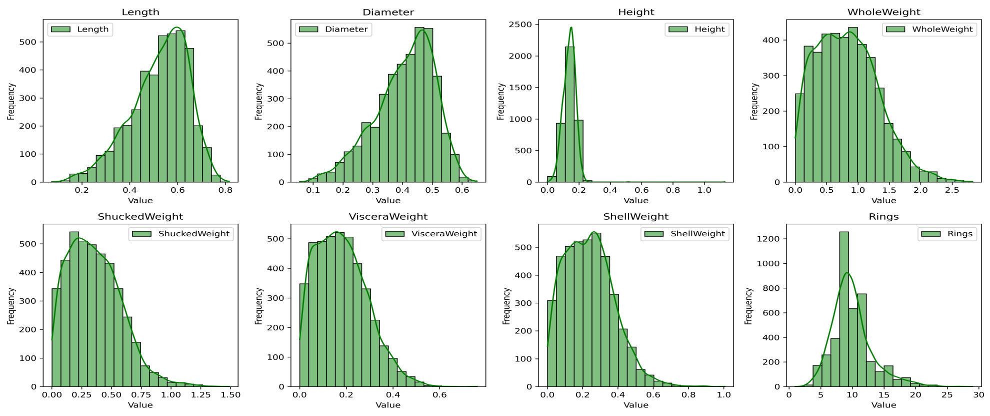

Figure 4: Distribution of Numerical Variables

Problem Statement

Predicting abalone age from physical measurements can be challenging. I will apply three ensemble machine learning methods: Boosting, Bagging and Random Forest, to predict abalone age using the Abalone dataset.

The goal of this analysis is to compare how well these three models predict abalone age based on physical measurements. I will use cross-validation and common performance measures, mean squared error (MSE), to assess their strengths and weaknesses. In the end, this study aims to find the most effective ensemble method for this task.

Method

Ensemble learning is a machine learning technique that combines multiple individual models (often called "weak learners") into one stronger model. Methods like Bagging, Boosting, and Random Forests work by pooling predictions to reduce errors and improve accuracy. If one model makes a mistake, it can be auto-corrected by others, making the final model more reliable.

(i) Gradient Boosting
Gradient Boosting improves predictions by training weak learners sequentially, with each new model correcting the errors of the previous one. For regression tasks, it starts with a simple model $F_0(x)$ (usually a constant) that has high bias and low variance. At each iteration m, the algorithm computes the residual $y - F_{m-1}(x)$ using the squared error loss and fits a new tree $h_m(x)$ to these errors. The model is updated as:

$$F_m(x) = F_{m-1}(x) + \nu\, h_m(x),$$

where ν is the learning rate. After M iterations, the final model is:

$$F_M(x) = F_0(x) + \nu \sum_{m=1}^{M} h_m(x).$$

This iterative process gradually reduces bias while keeping the model general enough to perform well on new data.

(ii) Bagging
Bagging, which stands for Bootstrap Aggregating, the process begins by creating multiple bootstrap samples from the training data with replacement, ensuring that each sample is independent of the others. A separate model is then trained on each bootstrapped dataset, and these models can be built in parallel. For regression tasks, the final prediction is obtained by averaging the outputs of all the models, which is mathematically represented as:

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$$

Here, $\hat{f}^{*b}(x)$ is the prediction from the b-th bootstrapped training set, and the final result is the average of all B models. This approach reduces variance while maintaining accuracy, leading to a more consistent and reliable model.

(iii) Random Forest

Random Forest builds many decision trees using random subsets of data and features, then averages their predictions for the final result. Each tree is grown on a bootstrap sample, and at every split, only a random set of features is used. This means that each tree has low bias but high variance, which helps avoid overfitting. By averaging the predictions across all trees in regression tasks, the overall variance is reduced, leading to a more stable model.

I deployed these three algorithms using grid search with 5-fold cross-validation to identify the optimal hyperparameters for each model. To ensure a fair comparison, I applyed the similar parameter grid across all models; for example, I tested the number of trees with values $[500, 1000, 2000]$ and maximum depth with values $[3, 5, 7]$. This consistent setup allows us to evaluate each model under similar conditions and directly compare their performance. Moreover, using 5-fold cross-validation ensures that every data point is used for both training and validation, making our results more reliable and providing clear insights into each model's effectiveness.

Model	**Parameter Grid**	**Optimal Parameters**
Gradient Boosting	"n_estimators": [500, 1000, 2000] "learning_rate": [0.01, 0.1, 0.2] "max_depth": [3, 5, 7]	"n_estimators": 1000 "learning_rate": 0.01 "max_depth": 3
Bagging	"n_estimators": [500, 1000, 2000] "estimator_max_depth": [3, 5, 7] "max_samples": [0.5, 1.0]	"n_estimators": 1000 "estimator_max_depth": 7 "max_samples": 0.5
Random Forest	"n_estimators": [500, 1000, 2000] "max_depth": [3, 5, 7] "max_samples": [0.5, 1.0]	"n_estimators": 1000 "max_depth": 7 "max_samples": 0.5

Table 1: Hyperparameter Grid and Optimal Parameters across Models

Results and Discussion

From the performance table (Table 2), Bagging and Random Forest both achieve a lower MSE (4.85) than Gradient Boosting (5.07), and they also run more than twice as fast. The fact that Random Forest and Bagging have the same MSE suggests similar predictive performance, but Random Forest runs slightly faster (3 min 41 s vs. 3 min 58 s). Meanwhile, Gradient Boosting takes the longest (9 min 19 s) and still yields a marginally higher MSE. Overall, Random Forest is the most efficient model, offering the best balance between accuracy and speed, while Gradient Boosting might require further tuning to improve performance.

Model	**Mean Squared Error**	**Execution Time**
Gradient Boosting: `best_gbr`	5.07	9m19'
Bagging: `best_bg`	4.85	3m58'
Random Forest: `best_rf`	4.85	3m41'

Table 2: Comparison of Model Performance

The Predicted vs. Actual plot (Figure 5) shows that, for lower and mid-range values, the predictions from all three models are close to the diagonal line, indicating a better match with the actual values. However, for higher ring counts (above about 15 rings), all models tend to underestimate, with the predicted values falling behind as the actual count increases. The Residual plot (Figure 6) confirms this trend, showing negative residuals around 10 rings and a positive drift beyond 15 rings. This suggests that the models could benefit from more data in the higher ring range, different hyperparameter tuning, or additional features to better capture extreme values.

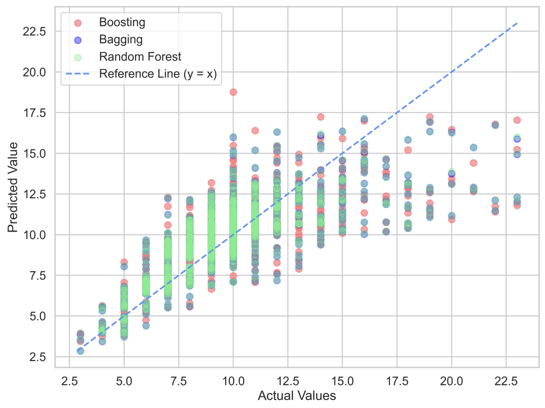

Figure 5: Predicted vs. Actual Value Plot

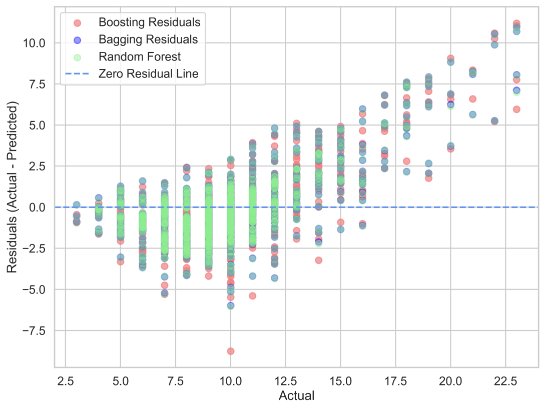

Figure 6: Residuals Plot

Conclusion

In conclusion, all three ensemble methods (boosting, bagging, and random forest) delivered moderate performance in predicting abalone ring counts. Bagging and random forest both achieved an MSE of 4.86, outperforming boosting which is 5.07. Moreover, random forest delivered the highest performance while being the fastest in computation. Residual analysis showed that all models tended to underestimate higher ring counts, suggesting that more data or additional features might be needed to improve performance in that range. Overall, bagging and random forest appear to be the best options for this dataset due to their accuracy and speed.