Prediction of Abalone Age

William Luo, Yihan Diao, Cheng Yang, Heqi Zhao, and Yushi Huang

***Abstract*—Abalone commands premium value in both aquaculture and the marine industry, and older specimens are especially valuable. The traditional way of determining abalone age is to count the rings on the inner shell surface—a costly, time consuming process that requires skilled workers. Our paper introduces an alternative method: using machine learning models to predict abalone age based on individual physical characteristics. To achieve this goal, we first analyze the summary statistics (mean, median, standard deviation, and skewness) of each feature to reveal underlying patterns in the abalone dataset provided by the Kaggle competition. Outliers are removed using the IQR method, and the categorical variable is one hot encoded. OpenFE is then used to create new incorporated features, and the top 15 features are selected from both original and newly generated features. Five different models—decision tree with XGBoost, CatBoost, linear regression, Random Forest, and a neural network—are trained and evaluated under 5 fold cross validation. The linear regression model and neural network have high MSE and low R^2. Among the remaining three models, XGBoost performs best, achieving R^2 of 0.6547 and RMSLE of 0.1464, which is very close to the current Kaggle leaderboard's top score of 0.14374. We have integrated this XGBoost model into a browser based interface that allows workers to estimate age by entering an abalone's characteristics and obtaining results accurately and rapidly. These results show that gradient boosted trees provide a practical, useful tool in commercial fisheries.**

***Index Terms*—Abalone, age prediction, feature engineering, regression, machine learning.**

I. Introduction

Abalone is a type of marine gastropod mollusk characterized by its single shell, herbivorous diet and usually lives in rocky reef habitats. It is widely distributed in temperate and tropical coastal water. Researchers have identified over 100 species of abalone so far and found the larger species usually are found in temperate areas while smaller species tend to inhabit tropical environments. Although all Abalone species have economic value, only about 20 types of abalone, relatively larger species, are considered commercially significant. The muscle tissue of abalone contains bioactive compounds with potential antioxidant and anticancer properties, contributing to its popularity among marine markets.[1]

The growth rate of abalone is seriously influenced by environmental conditions, especially for the strength of the water current and movement of waves. Due to the variation in food supply, the abalone living in the calmer water tends to grow slower than the abalone inhabited in more turbulent reef environments.[2] It's challenging to accurately estimate the age of abalone, as its size is not only dependent on age but also highly related to food supply. Given the closer relationship between abalone's age and commercial prices, it's meaningful and practical to precisely determine the age of abalone. In the past, one common age estimation method was to count the growth rings found inside the shell, which is similar to tree rings. The process requires people to slice the shell, polish and stain it, and then examine it under a microscope. This process is costly and laborious, leading to high market price and rarity of abalone.[3]

To address this age-estimation problem, machine learning algorithms have become a promising alternative by learning from existing data. Kaggle offers a dataset containing abalone ages and various physical features. This paper aims to predict an abalone's age based on its physical characteristics—such as sex, length, diameter, and height. First, researchers perform exploratory data analysis (EDA) to remove outliers and standardize the data scale, preventing model overfitting. Next, they use OpenFE—a tool that constructs, evaluates, and ranks newly derived features from the original feature pool—for feature selection. Only features exhibiting strong correlations with the target age are retained for model training.[4] By comparing the predictive accuracy of linear regression, neural networks, XGBoost (decision-tree boosting), and CatBoost (gradient boosting) algorithms, researchers identify the best-performing model. Finally, the selected model is integrated into a web-based front end that allows users to enter an abalone's physical characteristics and directly obtain its estimated age in real time.

May 24, 2025

II. Literature review

Much published literature has developed many useful predictive models similar to our topic—predicting outcomes based on numerical and categorical variables. However, relatively few papers predict abalone's age using machine learning. For example, Jabeen, K. and Ahamed, K. aim to develop an NLP model to predict abalone age.[5] This model utilizes the Levenberg–Marquardt algorithm to train the data. Their results suggest that as the number of hidden layers increases, the model's error rate gradually decreases, indicating that the model is robust and effective for prediction. Similarly, Misman et al. propose an artificial neural network based on linear regression to estimate abalone's age.[6] This structure is relatively shallow, with only three hidden layers. It is based on the physical characteristics of abalone, which is consistent with our topic. Sahin, E. and Saul apply a CNN model to this predictive task.[2] The authors experiment with various network structures and different types of convolution operations. Their findings demonstrate that machine learning approaches outperform traditional regression models in terms of accuracy.

We also consulted several studies that employed machine learning methods with similar structured datasets. For example, Tianqi Chen and Carlos Guestrin (2016) introduce XGBoost , a extensible decision tree boosting algorithm.[7] This algorithm can handle sparsity and missing values in real data for large-scale data on industrial scale, improving efficiency for large datasets. In additional, in the article ”Analysis and Classification of Heart Rate Using CatBoost Feature Ranking Model”, authors developed a machine learning model to classify heart rate condition based on different numerical value with CatBoost algorithm.[8] Although the domain differs, their datasets is consistent to ours, since we also need to predict categorical variables based on numerical features. This algorithm can also provide insight into which feature most significantly contributed to accurate classification.

Based on review of the literature and the needs of our paper, we decide to use linear regression, Xgboost(Decision Tree Boosting) and CatBoost(Gradient Boosting) algorithms to develop predictive model,aiming to balance interpretability, robustness, and predictive power.

III. EDA and Dataset Description

A. General information about dataset

The abalone dataset contains information on 90,614 abalone specimens. Including our target label *rings* and other attributes. **Sex**: represents the gender of the abalone, categorized as male (M), female (F), or infant (I). **Length**: The length feature denotes the longest measurement of the abalone shell, from the apex to the base. Range from 0.0750 to 0.8150 millimeters. **Diameter**: represent the measurement of the abalone shell perpendicular to its length. Range from 0.0550 to 0.6500 millimeters. **Height**: signify the height of the abalone shell, measured perpendicular to the plane formed by the length and diameter. Range from 0.0000 to 1.1300 millimeters. **Whole weight**: indicate the total weight of the abalone, encompassing both the meat and the shell. Range from 0.0020 to 2.8255 grams. **Whole weight.1** (Shucked weight): represent the weight of the abalone meat only, measured in grams. It indicates the amount of meat extracted from the shell. Range from 0.0010 to 1.4880 grams. **Whole weight.2** (Viscera weight): signify the weight of the abalone gut after bleeding, measured in grams. It provides insights into the weight of the internal organs of the abalone. Range from 0.0005 to 0.7600 grams. **Shell weight**: represent the weight of the abalone shell only, excluding the meat, measured in grams. It provides insights into the weight distribution between the shell and the edible portion of the abalone. Range from 0.0005 to 0.7600 grams.Target Labe **Rings** signify the number of rings present on the abalone shell. Range From 1 to 29. The following figure shows the mean, median and standard deviation of each feature.

TABLE I: Summary Statistics of Numerical Features

Feature	**Mean**	**Median**	**Standard Deviation**
Length	0.5171	0.5450	0.1182
Diameter	0.4017	0.4250	0.0980
Height	0.1355	0.1400	0.0380
Whole weight	0.7890	0.7995	0.4577
Whole weight.1	0.3408	0.3300	0.2044
Whole weight.2	0.1694	0.1660	0.1009
Shell weight	0.2259	0.2250	0.1302
Rings (Target)	9.6968	9.0000	3.1762

The table shows that the mean and median of most features are very close, such as Shell weight (mean 0.2259, median 0.2250) and Height (mean 0.1355, median 0.1400), indicating that the distribution of these variables is relatively symmetrical. Whole weight.1 (Shucked weight) is slightly left-skewed because its median is slightly smaller than the mean. The standard deviation of Whole weight is 0.4577, which is the most volatile feature except for the target variable Rings, indicating that this feature is more dispersed in the sample. Similarly, the standard deviation of Rings is 3.1762, which is much higher than other features, indicating that the distribution range of rings is wide.

B. Outlier Detection and Removal

In order to reduce the influence of extreme values on model training, we eliminated outliers for all numeric features except the sex feature. We used the IQR (interquartile range) method to define outliers. To be more specific, the 1st quartile (Q1) and 3rd quartile (Q3) for each numeric feature are first calculated, and then the outlier range is determined according to the formula:

$$x < Q_1 - 1.5 \times \mathrm{IQR} \quad \text{or} \quad x > Q_3 + 1.5 \times \mathrm{IQR}$$

We use box plots to visualize distributions , central tendency, and extreme values along with the calculated IQR-based thresholds. The following figure shows these attributes. All samples that fall outside of the above interval are deleted. After cleaning the dataset, about 3.7% are removed, leaving with 87267 rows of data

From the figure below, it can be observed that most features contain a certain number of outliers. In particular, *Whole weight*, *Whole weight.1*, and *Whole weight.2* exhibit significant outliers, especially above the upper bound. In contrast, features such as *Length* and *Diameter* have smaller and less extreme outliers.

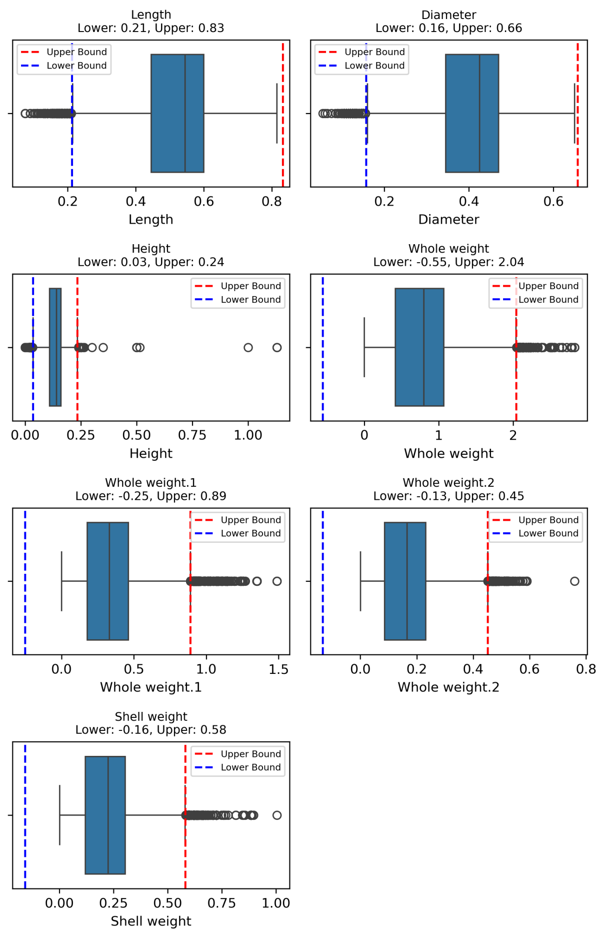

Fig. 1: Box plots of all numeric features with IQR-based outlier bounds

C. Distribution and One-Hot Encoding of Sex Feature

The image below shows the distribution of feature Sex in the dataset.The class 'I' (infant) represents the largest proportion at 36.1%, followed by 'M' (male) at 34.4% , and 'F' (female) at 29.5%. The figure indicates the balanced distributions of three different classes.

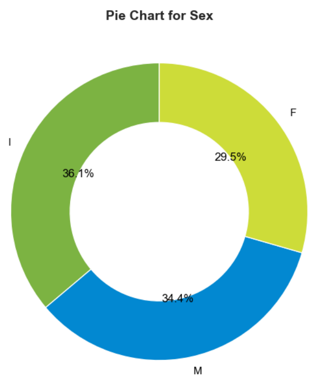

Fig. 2: chart showing the distribution of the *Sex* feature

One-hot coding is used to convert the sex label into a numeric feature for subsequent model processing. Attributes are converted to three new binary feature columns. The encoding rules are as follows:

TABLE II: One-Hot Encoding of the *Sex* Feature

Original value	Encoded Value
M	100
F	010
I	001

D. Skewness and distribution

Distribution analysis of all original numeric features in the training set was analyzed using histograms and skewness indicators. The results show that most features do not satisfy normal distribution and exhibit different degrees of skewness. To be more specific, features such as *Whole weight*, *Whole weight.1*, and *Whole weight.2* show right-skewed distributions, indicating the presence of long tails towards higher values. In contrast, *Shell weight* and *Shucked weight* satisfy the normal distribution, while *Length* and *Diameter* demonstrate slight left-skewness. Since we mostly use tree-based models, which are not sensitive to the distribution shape and numerical scale of the features, we did not perform normalization or standardization on the features except *Whole weight*.According to the models' actual performance on the dataset, applying a logarithmic transformation to *Whole Weight* can slightly improve the models' performance. Although *Whole weight.1* and *Whole weight.2* show more severe skewness, applying the same transformation leads to a decrease in performance, which may be due to the information structure in the original values being disrupted.

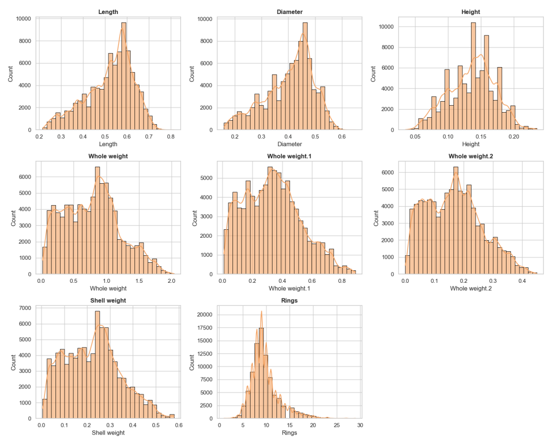

Fig. 3: Distribution of numerical features in the training set

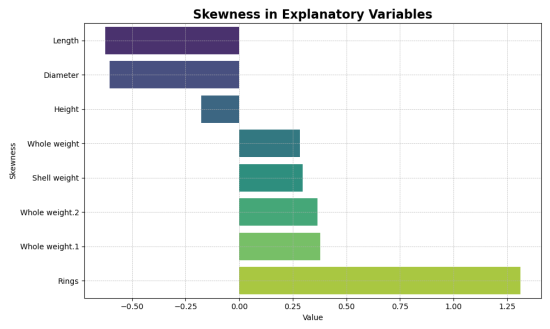

Fig. 4: Skewness values of numerical features

E. Feature Engineering with OpenFE

Our team uses the OpenFe tool to generate new combined features. We select the 11 most informative features generated by OpenFE from the full set of 217 features for the next round of filtering.

TABLE III: Top 11 OpenFE-Generated Features Selected for Further Filtering

Index	Feature Expression
1	Length / Shell weight
2	Whole weight.1 / Shell weight
3	Diameter / Shell weight
4	Whole weight / Whole weight.1
5	Length - Shell weight
6	Shell weight / Sex
7	freq(Shell weight)
8	max(Whole weight.2, Shell weight)
9	max(Shell weight, Height)
10	Shell weight + Height
11	freq(Whole weight)

we removed the feature *Shell weight / Sex* from the list first. Since the *Sex* variable was transformed using one-hot encoding, it no longer represents a comparable numerical value. As a result, arithmetic operations such as division involving this variable are not meaningful.

F. Final Feature Selection Using Correlation, Mutual Information, and Scatter Plots

Pearson correlation measures the linear correlation between two continuous variables, with a range from -1 to +1.

$$r_{X,Y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

The heatmap shows the Pearson correlation coefficients between each feature. It can be observed that the correlation between *Shell weight*, *Whole weight*, *Length*, *Diameter*, *Height*, and their derived features exceeds 0.9, indicating significant linear redundancy. The correlation between *freq(Shell weight)*, *Sex*, *freq(Whole weight)*, and other features, as well as the target variable, is very low, which may indicate they are independent variables or contain noise.

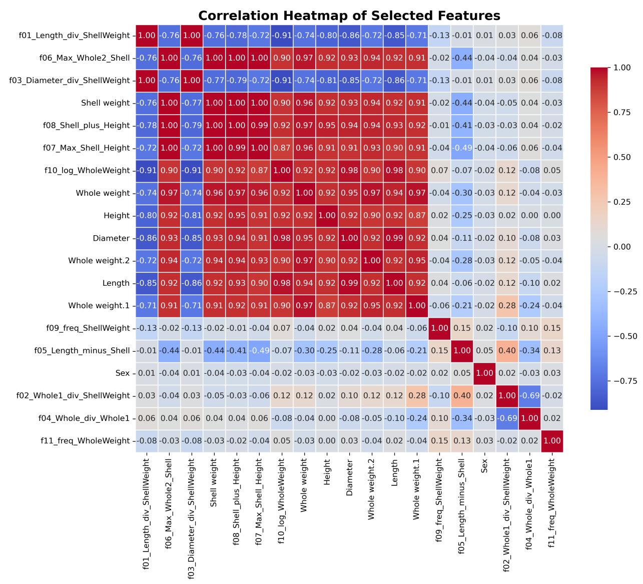

Fig. 5: Correlation heatmap of selected features including original and OpenFE-generated variables

Mutual information is defined in terms of entropy. The entropy $H(X)$ of a random variable X represents the average amount of information or uncertainty in X, it is defined as:

$$H(X) = -\sum_{x\in X} p(x)\log p(x)$$

The mutual information between two variables X and Y is defined as:

$$I(X;Y) = H(X) - H(X \mid Y)$$

It can also be expressed as the disjoint probability:

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

we use mutual information to evaluate how much information each feature provides about the target variable *Rings*

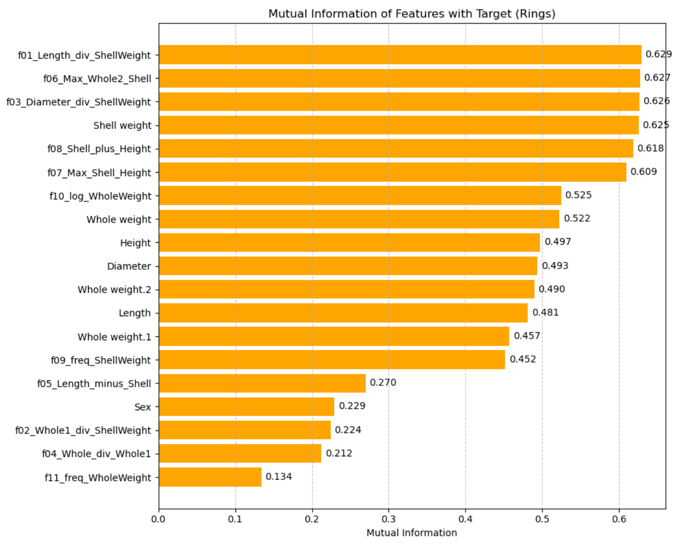

Fig. 6: Mutual Information Scores between each feature and the target variable *Rings*

Based on the MI score, we found that most combined features exhibit a stronger correlation with the target variable. In the original features, weight-related variables perform well.

Several features are removed after the combined analysis of the correlation heatmap and mutual information scores. First, the feature *Whole weight / Whole weight.1* was removed due to its relatively low mutual information score, indicating limited predictive value. Furthermore, we drop the feature *Max(Shell, Height)*, which showed a high correlation with many other features in the heatmap, but had a lower MI score compared to *Shell weight + Height*. To reduce redundancy and the risk of overfitting, we only keep the more informative one. Furthermore, the original characteristic *length* was removed, since it was found to be highly correlated with other size-related variables. Finally, *Whole weight* was replaced with its logarithmic transformation *log(Whole weight)*, as previously justified in the Skewness and distribution section. The table below presents the final set of selected features. The frequency feature represents the proportion of times that a certain value appears in the dataset.

TABLE IV: Final Selected Features for Modeling

Number	Feature
1	Diameter
2	Height
3	Whole weight.1
4	Whole weight.2
5	Shell weight
6	Sex
7	Length / Shell weight
8	Whole weight.1 / Shell weight
9	Diameter / Shell weight
10	Length - Shell weight
11	Frequency of Shell weight
12	Max(Whole weight.2, Shell weight)
13	log(Whole weight)
14	Frequency of Whole weight
15	Shell weight + Height

We finally created a pairwise scatter plot to identify trends between selected features and rings to give us an idea of selecting models. As can be seen from the figure, a small number of features (such as Shell weight, Whole weight, Diameter, Height) have a concentrated distribution trend and linear characteristics. Most of the constructed features (such as Length / Shell weight, Diameter / Shell weight, Length - Shell weight, etc.) show a more complex distribution pattern, and the relationship is non-linear. They are more suitable for nonlinear models, such as decision trees, XGBoost, and Random Forest.

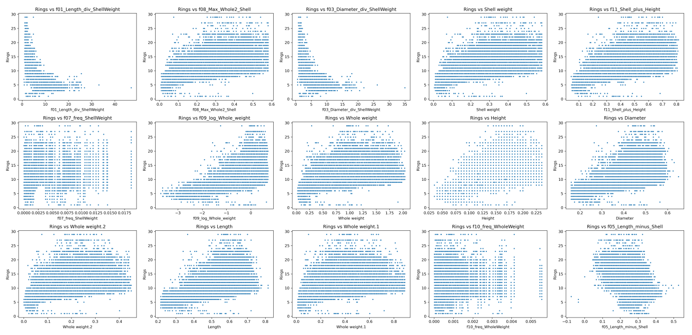

Fig. 7: Scatter plots of *Rings* vs selected features

IV. Proposed methodology

A. Project Design

Our project aims to predict the rings of abalone using supervised learning strategies. This is quantitative research. We select the most relevant features and use them to construct a new dataset. All models will be trained and evaluated on the same dataset to make sure that the performance comparison between different models is fair.

B. Data preprocessing

Firstly, we analyze some basic information about each feature in the original dataset, including mean, median, standard deviation, skewness, and physical meaning. Second, we use the interquartile range (IQR) method to remove outliers in our dataset, along with applying one-hot encoding on the non-numeric attribute *Sex*. Third, the cleaned dataset is fed into OpenFE, an automated tool for feature engineering, which generates 11 high-quality features that are helpful for prediction. With the 11 newly generated features and 8 original features, we applied mutual information scores and correlation analysis to carry out the final feature selection. As a result, four features were dropped due to high correlation with others to reduce overfitting, or limited contribution to predicting the target variable, *Rings*.

C. Model development

We plan to use various models for our prediction task, including regression algorithms, decision trees, and neural networks. For each model, we apply Optuna, an intelligent optimization agent for hyperparameter tuning. It applies Bayesian optimization algorithms to dynamically explore the search space. Compared to other methods, Optuna is more adaptive and efficient in learning optimal parameters. Moreover, we apply 5-fold cross-validation to help determine the best hyperparameter combinations. The dataset is divided into training and testing sets with an 80:20 split.

1) Decision Trees: For the decision trees strategy, three algorithms are applied- XGBOOST, CATBOOST, and Random Forest. Decision trees can deal with complex non-linear data structures and do not require normalization of numerical features. The parameter ranges used for tuning each model are shown below.

- XGBOOST

Parameter	Range	Description
n_estimators	50 – 500	Number of boosting rounds
learning_rate	0.01 – 0.3	Step size
gamma	0 – 1	Minimum loss reduction
reg_alpha	0 – 1	L1 regularization term on weights
reg_lambda	0 – 2	L2 regularization term on weights
max_depth	4 – 20	Maximum depth of a tree
min_child_weight	1 – 10	Minimum sum of instance per child
subsample	0.6 – 1.0	Training sample fraction per round
colsample_bytree	0.6 – 1.0	Fraction of features used per tree

TABLE V: Hyperparameter search ranges for XGBoost

- CatBOOST

Parameter	Range	Description
iterations	100 – 500	Number of boosting iterations
learning_rate	0.01 – 0.3	Step size (learning rate)
depth	4 – 10	Depth of each tree
l2_leaf_reg	1 – 10	L2 regularization term
random_strength	1e-9 – 10.0	Random noise for split scoring
temperature	0 – 1.0	reflect diversity of sampled trees

TABLE VI: Hyperparameter search space for CatBoost

- Random Forest

Parameter	Range	Description
n_estimators	100–500	Number of trees in the forest
max_depth	4–30	Max depth of each decision tree
min_split	2–10	Min samples to split a node
min_leaf	1–10	Min samples at a leaf node
max_features	sqrt, log2	Number of features for each split

TABLE VII: Hyperparameter search space for Random Forest

2) Linear Regression: For linear regression, we applied three different training methods: batch gradient descent (BGD), mini-batch gradient descent, and stochastic gradient descent (SGD). Due to the large size of the dataset, the model is expected to take longer to converge when using SGD. We set the learning rate to 0.015 for BGD and SGD, and 0.001 for mini-batch gradient descent. Each model was trained for 100 epochs, and the MSE trend over epochs was visualized. The model with the lowest MSE among the three training methods will be selected for comparison with other models.

3) Neural Networks: We built a neural network with two hidden layers, using the ReLU as the activation function for each neuron. The Optuna algorithm is applied to search for the optimal combination of hyperparameters, with a total of 25 trials conducted during the search. The parameter ranges used for tuning each model are shown below.

Parameter	Range	Description
hidden_size	16 – 64	Number of neurons in each hidden layer
dropout_rate	0.3 – 0.7	Dropout rate for regularization
batch_size	256 – 1024	Batch size for training
lr	0.001 – 0.1	Learning rate

TABLE VIII: Hyperparameter ranges for neural network

D. Evaluation Method

The values of MSE, R-squared, and RMSLE will be calculated to evaluate the performance of the model. Additionally, a plot of MSE against the number of epochs (for neural networks and linear regression models) or the number of estimators (for decision trees) will be generated to visualize the training trend. At the same time, the model's RMSLE score will be compared with the competition results of other teams on Kaggle. Among them, the highest RMSLE score of the competing teams is 0.14374.

V. Experimental results and evaluation

For each model, we train with the optimal hyperparameter combination searched by Optuna, and apply 5-fold cross-validation to get assessment indicators such as MSE, RMLSE, and R^2. We plot the graph of Mean Squared Error (MSE) as it varies with the n estimator on both the training set and test set.

A. Decision Trees

1) XGBOOST: The best hyperparameter combinations for XGBOOST are shown in the table below.

Parameter	Best Value
n_estimators	356
learning_rate	0.0182
gamma	0.7311
reg_alpha	0.6105
reg_lambda	0.2070
max_depth	9
min_child_weight	8
subsample	0.6387
colsample_bytree	0.6987

TABLE IX: Best Hyperparameters for XGBoost Model

The value of n_estimators is 356, which implies that the model has relatively high complexity and can capture more intricate patterns in the data. The learning rate is set to a small value of 0.0182, indicating that the contribution of each individual tree is relatively small. The parameters gamma (0.7311), reg_alpha (0.6105), and reg_lambda (0.2070) are used to regularize the model and help prevent overfitting. The max depth(9) and min_child_weight(8) determine the depth and conditions for splitting a node. The subsample(0.6387) and colsample_bytree(0.6987) determine the proportion of random samples and the portion of random characteristics for each tree. The figure below shows the trend of MSE as the n_estimators(number of decision trees) increase.

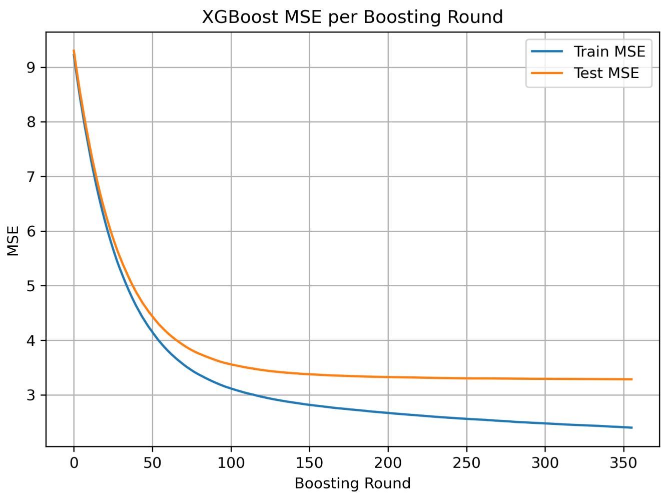

Fig. 8: MSE curves of training and test sets

As can be concluded from the figure, the value of MSE on the training set decreases rapidly when the number of trees is from 0 to 100. After the number of trees exceeds 100, MSE tends to decrease slowly until convergence. The following table shows the exact value of assessment indicators, including MSE, R^2, and RMLSE, on the test set.

Metric	Value
Test MSE	3.2855
R^2	0.6547
RMSLE	0.1464

TABLE X: Performance metrics of the XGBoost model on the test set

2) CATBOOST: The best hyperparameter combination for CATBOOST is shown in the table below.

Parameter	Value
iterations	498
learning_rate	0.1076
depth	7
l2_leaf_reg	8.2323
random_strength	0.5455
bagging_temperature	0.9923

TABLE XI: Best hyperparameters and performance metrics of the CatBoost model

Iteration (498) determines the number of trees, indicating that the complexity of the model is very high. learning_rate (0.1076) is a large value, which greatly increases the convergence speed. The depth of the tree is 7, which means that it can capture features of medium complexity and can handle nonlinear relationships. l2_leaf_reg (8.2323) is the regularization coefficient, which is a larger value that helps reduce the risk of overfitting. bagging_temperature (0.9923) controls the diversity of samples. The maximum value is 1, so the diversity of the model is very high. random_strength (0.5455) indicates that randomness is added in the process of constructing trees. The figure below shows the trend of MSE as n_estimators(number of decision trees) increases.

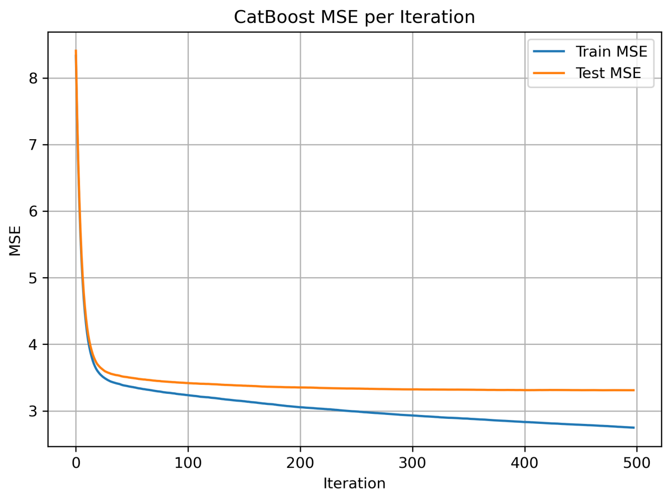

Fig. 9: MSE curves of training and test sets for CatBoost

As can be seen from the figure, the value of MSE decreases very quickly in the first 20 to 30 trees due to the large learning rate. After that, the value of MSE decreases slowly until convergence. The following table shows the exact value of assessment indicators, including MSE, R^2, and RMLSE on the test set.

Metric	Value
Test MSE	3.3096
R^2	0.6521
RMSLE	0.1471

TABLE XII: Performance metrics of the CatBoost model on the test set

3) Random Forest: The best hyperparameter combinations for Random Forest are shown in the table below.

parameter	value
n_estimators	390
max_depth	19
min_samples_split	9
min_samples_leaf	2
max_features	sqrt

TABLE XIII: best hyperparameter combination

The number of trees is 390, indicating that the model requires a large number of trees to capture complex features and enhance generalization ability. The depth of the trees (19) suggests a strong dependency between features. min_samples_split (9) indicates that trees will only split when the sample size is sufficient, effectively preventing overfitting. min_samples_leaf (2) indicates that each leaf node contains very few samples, which is beneficial for capturing rare features. max_feature uses the ”sqrt” function to reduce the number of features used for splitting nodes, effectively preventing overfitting. The figure below shows the trend of MSE as n_estimators(number of decision trees) increases.

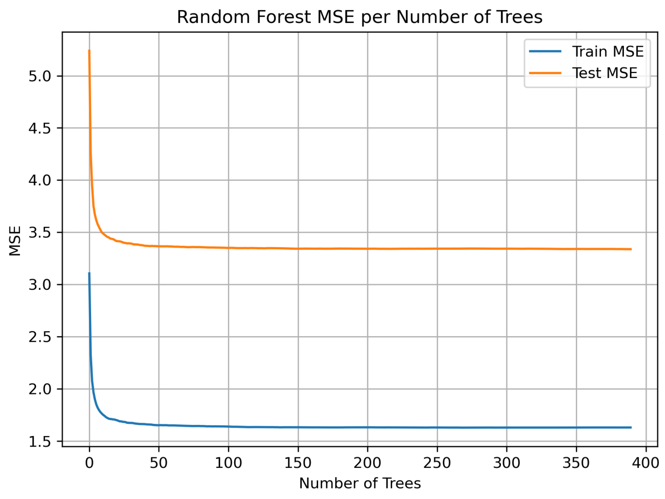

Fig. 10: MSE curves of training and test sets for Random Forest

Unlike two other decision tree models, the performance of the random forest model varies greatly between the training set and the testing set, showing significant overfitting. The MSE value decreases rapidly in the first 10 trees, then slows down until it stabilizes. This aligns with the characteristics of the random forest model: able to capture the most obvious features and rectify bias quickly. The following table shows the exact value of assessment indicators, including MSE, R^2, and RMLSE, on the test set.

Metric	Test Set Performance
MSE	3.3383
R^2	0.6491
RMSLE	0.1475

TABLE XIV: Main regression metrics of the Random Forest model on the test set

B. Neuron Network

Our neural Network has two hidden layers, and for each hidden layer, we use ReLU as an activation function to help capture complex characteristics. Since our project is a regression task, we use a linear activation function in the output layer, allowing the model to predict any real number. The best hyperparameter combinations for Neural Network are shown in the table below.

Hyperparameter	Optimal Value
Hidden size	61
Dropout rate	0.327
Batch size	541
Learning rate	0.0010

TABLE XV: Optimal hyperparameters for the ANN regression model

Hiddensize(61) indicates that the number of neurons in each hidden layer is 61. This is a moderate number that can balance capturing nonlinear relationships and avoiding overfitting. The dropout rate (0.327) is a moderate value that can effectively prevent overfitting. The batch size (541) is a relatively large number, which can speed up learning but may also capture more noise. The learning rate (0.0010) is a very small value, which ensures stable convergence. The figure below shows the trend of MSE as epoch increases on both the training and test sets.

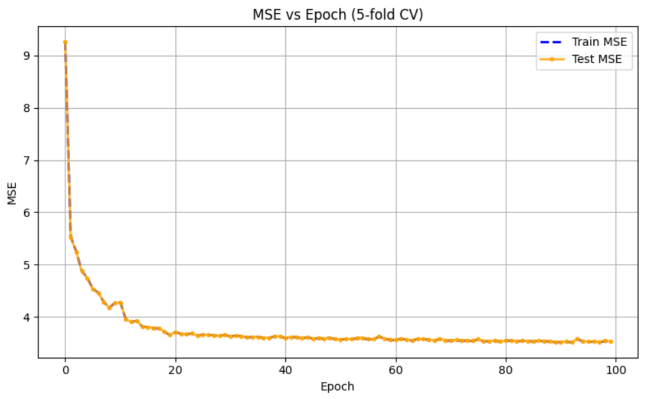

Fig. 11: MSE curves of training and test sets for the ANN model

It can be seen from the figure that the model's performance on the training set and the test set is close, which means that there is no occurrence of overfitting or underfitting. The MSE decreases rapidly in an oscillatory pattern during the first 20 epochs, and then decreases slowly with reduced oscillation until convergence. The following table shows the exact value of assessment indicators, including MSE, R^2, and RMLSE on the test set.

Metric	Value
MSE	3.5304
R^2	0.6267
RMSLE	0.1535

TABLE XVI: ANN-test-metrics

C. Linear Regression

The best hyperparameter combination for LinearRegression is shown in the table below.

Parameter	Value
iterations	100
batch_size	32
learning_rate	0.0008
alpha	0.001

TABLE XVII: Best hyperparameters and performance metrics of the LinearRegression model

The Linear Regression model represents the simplest approach to single-feature regression. We are selecting the best performance from three gradient descent that are SGD, BGD, and mini-BGD. We are implementing this function by passing the different parameters, like batch_size, to determine the gradient descent function we want. It is aligned with the speculation we made, SGD has a lower convergence speed and very obvious oscillation in MSE changing over epochs. Once we switch to the mini-batch gradient descent, the curve gets smoother. We still use the same methods of recombining the features that are highly correlated to improve the quality of the dataset. To search for the best parameter combinations, we applied the GridSearchCV. We put the parameters in a dictionary and train and test each combination of those parameters, then extract the model with the lowest MSE score and return the parameters. The figure below shows the trend of MSE per iteration.

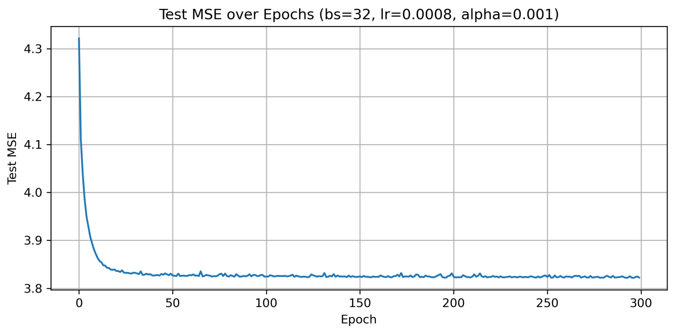

Fig. 12: MSE curves of training and test sets for Best Linear Regression

As we can see from the figure, the value of MSE decreases suddenly in the 20 epochs, and becomes very stable and has a little oscillation in the following training epochs, which is a feature of mini-bgd. So it converges very quickly and with a fair result. Since linear Regression only captures the linear relationship between the feature and predicted value, it is definitely not enough in this situation. So it has a much higher MSE value than XGBoost and Catboost we have trained earlier.

Metric	**Value**
Test MSE	3.8224
R^2	0.5982
RMSLE	0.1589

TABLE XVIII: Performance metrics of the Linear Regression model on the test set

D. Comparison Models

In this section, we evaluate the performance of each machine learning algorithm. The table below shows the performance of each model on the test set.

Model	**Test MSE**	R^2	**RMSLE**
XGBoost	3.2855	0.6547	0.1464
CatBoost	3.3096	0.6521	0.1471
Random Forest	3.3383	0.6491	0.1475
Neural Network	3.5304	0.6267	0.1535
Linear Regression	3.8224	0.5982	0.1589

TABLE XIX: Performance comparison of five models on the test set

From the figure, it can be observed that the linear regression performs as a baseline model with the lowest R^2 score and the highest MSE. The reason is that linear regression assumes that all features satisfy the normal distribution. But we only normalized the **whole_weight** among the 15 features. This is a deliberate trade-off because normalizing all features weakens the performance of decision tree-based models, and decision tree models themselves are not sensitive to feature scaling. Furthermore, the linear regression model is not sufficient to capture a large number of complex features.

The performance of the ANN model is not very good, with an MSE of 3.5. The reasons for poor performance should be similar to those of the linear model. ANN is sensitive to the scale of features. However, we did not standardize all skewed features. In addition, our training has encountered the problem of overfitting. We think this is due to the excessive and complex number of features. Moreover, we should try some new activation functions or optimizers for the parameter tuning process.

The performance of the three models based on the decision tree is excellent. XGBoost achieved the lowest MSE (3.2855), followed by CatBoost (3.3096) and Random Forest (3.3383). They perform well because we use OpenFE to generate some new features with strong expressive ability, which significantly enhances the performance of the decision tree-based model. Moreover, tree models are very good at learning complex nonlinear relationships between features.

The winning model is XGBOOST. XGBoost uses Lasso and Ridge regularization, which helps to prevent overfitting. Its optimization method is based on the second derivative of the loss function, which is more refined, more quickly, and more accurately compared to the first-order gradient optimization method of CatBoost. Additionally, XGBOOST responds more positively to OpenFE features compared to Catboost and random forest. From the R^2 metric perspective, our best model, XGBoost, achieved 0.6547, indicating that the model can explain about 65.5% of the label variance, with a moderate fitting result. For RMLSE, our XGBoost model achieved a score of 0.1464, which is not much different from the first-place score(0.14374) in the Kaggle competition, indicating that our best model is competitive.

VI. Conclusion

The paper proposes a machine learning based model to predict abalone age from individuals' physical traits. After removing data outliers and utilizing newly generated incorporated features, we compared the MSE, R^2, and RMSLE of five models and found that the decision tree model with XGBoost achieved the best prediction performance. We have integrated this tree model into a browser-based interface that enables workers to input an abalone's characteristics and obtain accurate age estimations. The limitation of the model is that the data lacks environmental variables. As the references mention, habitat, water temperature, and salinity can affect shell growth; consequently, the model may miss systematic differences between abalones from different habitats and rearing conditions. Although the tree-based algorithm mitigates scale sensitivity, linear regression and the neural network are negatively influenced by skewed weight features, which can be solved by logarithmic transformation or Box-Cox. Finally, an R^2 of 0.65 suggests that there is room to improve the model's fit. For future work, habitat and specimen information can be supplemented to capture growth differences among regions. We can apply logarithmic or Box-Cox transformations to weight variables to meet model assumptions. Investigation into stacking tree-based and neural models, or adopting newer tabular deep learning architectures, may also increase R^2 beyond 0.70.

VII. Github and Demo link and Project roadmap and team roles

This project was completed through close collaboration within our team, with each member taking on specific responsibilities. William and Yihan conducted the exploratory data analysis, including data examination and feature selection. Cheng developed the linear regression model, while Heqi worked on the CatBoost model. Yihan implemented the neural network, random forest, and XGBoost models. Yushi was responsible for developing both the front-end and back-end of the website, as well as recording the demo. All team members contributed to the final report. The project roadmap is shown in the table below:

Week	**Tasks**
Week 1	Topic selection and project planning
Week 2–4	Exploratory Data Analysis (EDA) and Feature Engineering
Week 5–7	Development of five models: Linear Regression, Random Forest, XGBoost, CatBoost, and Neural Network. Hyperparameter tuning
Week 8–9	Final report writing and demo video recording

TABLE XX: Project Roadmap

You can find the links to our GitHub repository and project demo video below:

- https://github.com/Dawn0115/ECS171-GroupProject
- https://youtu.be/n7MSVpWnjXk

References

[1] Khadija, *Abalone Age Prediction using Artificial Neural Network*, IOSR Journal of Computer Engineering, Vol. 18, pp. 34–38, September 2016. https://doi.org/10.9790/0661-1805023438

[2] M. F. Misman *et al.*, *Prediction of abalone age using regression-based neural network*, in *Proc. 1st Int. Conf. on Artificial Intelligence and Data Sciences (AiDAS)*, pp. 23–28, IEEE, September 2019. https://doi.org/10.1109/AiDAS47888.2019.8970983

[3] Zhengjie Wang, *Abalone Age Prediction Employing A Cascade Network Algorithm and Conditional Generative Adversarial Networks*, Research School of Computer Science, Australian National University.

[4] Tianping Zhang, Zheyu Zhang, Zhiyuan Fan, Haoyan Luo, Fengyuan Liu, Qian Liu, Wei Cao, Jian Li, *OpenFE: Automated Feature Generation with Expert-level Performance*, arXiv preprint arXiv:2206.14255, 2022. https://arxiv.org/abs/2206.14255

[5] K. Jabeen and K. Ahamed, *Abalone age prediction using artificial neural network*, IOSR Journal of Computer Engineering, vol. 18, no. 5, pp. 34–38, 2016. https://doi.org/10.9790/0661-1805023438

[6] E. Sahin, C. J. Saul, E. Ozsarfati, and A. Yilmaz, *Abalone life phase classification with deep learning*, in *Proc. 5th Int. Conf. on Soft Computing & Machine Intelligence (ISCMI)*, pp. 163–167, 2018. https://doi.org/10.1109/ISCMI.2018.8703232

[7] Tianqi Chen and Carlos Guestrin, *XGBoost: A Scalable Tree Boosting System*, arXiv preprint arXiv:1603.02754, 2016. https://arxiv.org/abs/1603.02754

[8] B. Dhananjay and J. Sivaraman, *Analysis and classification of heart rate using CatBoost feature ranking model*, Biomedical Signal Processing and Control, vol. 68, 102610, July 2021. https://doi.org/10.1016/j.bspc.2021.102610