Comparison of Tree-Based Methods and Ensembles for Classification and Regression Performance on Abalone Dataset

Tanay Sunil Panat
School of Mathematics and Science
University of New South Wales
Sydeny, Australia
t.panat@student.unsw.edu.au

***Abstract*— This project presents an algorithmic assessment of ensemble learning and tree-based approaches for classifying abalone into four age categories. While the age of abalone can be determined in laboratory, this can be a time-consuming process, we provide a fast approach for assessing its age using decision trees and ensemble learning methods. The results indicated that abalone age can be determined by its physical measurements with less time consumption. A pre-processing approach was performed to the dataset before constructing and training the model. We conducted various experiments using Decision Trees with pre-pruning, Random Forest Tree, XGBoost, Gradient Boosting and Simple Neural Network with ADAM optimizer. All the proposed models were trained on Abalone dataset procured from University of California at Irvine Machine Learning Repository. Of all the models applied, Extreme Gradient Boosting Algorithm yielded the highest accuracy of 66.0%, surpassing the rest of the models.**

***Keywords* – Ensemble learning, Decision Tree, pre-pruning, Random Forest Tree, XGBoost, Neural Network, ADAM optimizer, pre-processing.**

I. INTRODUCTION

Abalone is a mollusc with a single shell, herbivore aquatic gastropod which belongs to the family Haliotidae. Abalones are marine snails; they have been harvested worldwide for many decades as a source of food and decorative items [1]. The limited fishing seasons of abalone have elevated it to a epicurean dish. The age of the abalone plays a crucial role in determining its price, creating a proportional relationship that impacts both farmers and customers. [2]. Determining the age of abalone is essential as it helps the marine biologists understand abalones habitat and can implement measures to safeguard this species as these species are facing risk of extinction due to overfishing and acidification of oceans [1].

Age is determined by the number of rings present in the inner shell of the abalone, one ring is developed every year. To determine the age, we first split the abalone shell through the cone and then stain the shell which makes the rings prominent [3]. Then with the help of a microscope number of rings present in the abalone are counted. This process is cumbersome and time consuming for the farmers as well as the marine biologists. Hence, to improve the cost as well as time efficiency various machine learning and deep learning models can be implemented.

To resolve the issue of age detection we have implemented multiple models, but some stand out from the rest. Random Forest Classifier, this model has gained traction in recent years due to its ability of having low variance as well as low bias while predicting the class [4]. It outperforms decision tree because, decision tree has low bias and high variance, the training error will be less, high variance states that the error while testing will be high which means we can face the issue of overfitting. In case of random forest classifier, we have multiple decision trees to which some randomly selected features are fed via row sampling and feature sampling and the overall aggregation of these trees reduces the high variance as we take the majority vote of these decision trees for classification [4].

In the present study, along with random forest trees, we experimented with different pruning techniques as well as implemented various ensemble methods like XGBoost, Random Forest and Gradient Boosting which not commonly seen being implemented on this dataset. These techniques are capable of handling non-linearity in the data and we do not need to select features as feature selection are automatically performed by these techniques as well as due to their robustness to outlier noise. Considering the number of features as well as presence of outliers in the dataset, use of these techniques is justified.

In this project we will be investigating mainly 5 algorithms: Decision Trees, Random Forest, XGBoost, Gradient boosting and simple Neural Networks. In decision tree we will be looking at the effect of per-pruning on the accuracy of age classification and experimenting with hyperparameter mainly the “depth” of the tree and “criterion”. We will be conducting experiments with variable depth of Random Forest to observe its effect on the accuracy of classification. To conduct all the experiments, we will be using Abalone Dataset provided by the UC Irvine Machine Learning Repository [5]. This project is mainly a classification project, but we will also compare the results of classification models with linear regression model.

The remaining paper is organized as follows. In Section 2 some related work is described. In Section 3 methodology. Section 4 contains model development and performance analysis. Section 5 presents the results of experiments conducted and the paper is concluded in Section 6.

II. RELATED WORK

The classification and analysis of abalone age prediction had captured the interest of many researchers in the field of machine learning and marine biology. UC Irvine machine learning repository hosts a dataset containing various physical attributes such as their measurements, sex, weight, and age [5]. This dataset has served as the foundation for various research in the area focusing on species identification, age prediction, and growth pattern analysis.

Several noteworthy works have used the UCI abalone dataset to investigate and address various elements of abalone age classification. For instance, Harryt [6] conducted a comprehensive analysis using decision trees to perform classification of abalone age into 2 categories young (0-9) and old (10-29). They used post-pruning and online pruning on the training dataset and 10-fold-cross-validation was implemented to evaluate the performance of the classifier.

Muhammad Faiz Misman et al. (2019) [7], used ANN for prediction of abalone age. They used 3 hidden layer architecture, there were 11 nodes in the input layer, 6 nodes in the rest of the three hidden layers all these layers were having ReLu as their activation function and the output function had 1 node with linear activation function. They used RMSE value as their evaluation matric and to tackle the issue of overfitting dropout layers were used in the hidden layer. After doing PCA on the training data and implementing ANN they were able to achieve RMSE score of 2.2809.

Similarly, Egemen Sahin et al. (2018) [8] and Aalayya (2010) [7] used neural networks to predict the age of abalone. Egemen Sahin implemented 3 types of neural networks, Convolutional Neural Networks (CNN), Artificial Neural Network (ANN) and Residual Neural Network (RNN). Simple ANN with ReLu as activation function had the highest accuracy in comparison to CNN and RNN with different parameters.

Aalayya categorized the age into 4 categories group1 having abalones from 3-7, group2 having abalones from age 8-12, group3 having records aged between 13-17 and group4 with 18-21. They implemented network architecture having one hidden layer with eight input attributes.

In this project for the neural networks part, I have implemented ADAM optimizer which stands for Adaptive Moment Estimation. The fundamental task of optimizers is to update the weights and reduce the errors.

$$W_{new} = W_{old} - n * \partial \mathrm{L} / \partial \mathrm{W_{old}}$$

Where W_{new} is the updated weight, "n" is the learning rate and L is the loss function. ADAM optimizer is the fusion of Adaptive Gradient Algorithm and Root Mean Squared Propagation [9]. AdaGrad enhances performance on sparse gradient problems by maintaining a per-parameter learning rate. RMSProp, on the other hand, adjusts per-parameter learning rates based on the recent average magnitudes of weight gradients, reflecting the rate at which they change. This characteristic indicates the method's effectiveness in handling both online and non-stationary issues, such as noisy scenarios [9]. ADAM is considered the best optimizer.

ADAM optimizer uses the average of the second moments of the gradients (the uncentered variance). The algorithm computes an exponential moving average for both the gradient and the squared gradient, with the decay rates of these moving averages controlled by the parameters beta1 and beta2 [10].

Algorithm 1 Adam Optimization Algorithm

$V_{dw} = 0,\ S_{dw} = 0$
$V_{db} = 0,\ S_{db} = 0$
for iteration t **do**
  Compute $\frac{\partial L}{\partial w}, \frac{\partial L}{\partial b}$ using current mini-batch
  $V_{dw} = \beta_1 \cdot V_{dw} + (1 - \beta_1) \cdot \frac{\partial L}{\partial w}$
  $V_{db} = \beta_1 \cdot V_{db} + (1 - \beta_1) \cdot \frac{\partial L}{\partial b}$
  $S_{dw} = \beta_2 \cdot S_{dw} + (1 - \beta_2) \cdot \left(\frac{\partial L}{\partial w}\right)^2$
  $S_{db} = \beta_2 \cdot S_{db} + (1 - \beta_2) \cdot \left(\frac{\partial L}{\partial b}\right)^2$
  Bias Correction:
  $W_t = W_{t-1} - \eta \cdot \frac{V_{dw}}{\sqrt{S_{dw}+\epsilon}}$
  $b_t = b_{t-1} - \eta \cdot \frac{V_{db}}{\sqrt{S_{db}+\epsilon}}$
end for

This algorithm was adapted from ADAM optimizer paper written by kingma. [9]

In addition to Neural Networks, I have also implemented a decision tree. Pre-pruning and post-pruning are important concepts to increase the accuracy of decision tree by eliminating the branches from the tree which are not adding enough information relevance to the model [11]. I implemented pre-pruning in place of post pruning because pre-pruning is much quicker in comparison to post-pruning. It constrains the construction of the decision tree by considering enhancements in purity, signifying an increase in information gain for the Decision Tree operator and an improvement in the Gini coefficient for the CART operator.

In this project, the aim was to conduct a comparative analysis of various machine learning techniques for abalone age classification. The methods under scrutiny included decision tree classifiers, Random Forest, XGBoost, Gradient Boosting, and a Neural Network implemented with the Adam optimizer. Each method was assessed and compared based on their performance in predicting the age of abalones using the UCI abalone dataset. The goal was to provide a comprehensive understanding of the strengths and weaknesses of these diverse approaches in the context of abalone age prediction.

This comparative analysis not only contributed to the existing body of research but also offered insights into the most effective methodologies for accurate abalone age classification, thus enhancing the understanding and applicability of machine learning in marine biology and species classification studies.

III. Methodology

This section is divided into 5 subsections: A) Data Pre-Processing and Analysis, B) Model Overview, C) Framework and Workflow Diagram, D) Software Prerequisites and E) Experimental Setup and Reproducibility Framework. The first section will give analytical insights as well as pre-processing done for the data to be used in the machine learning models. The second section will discuss the experimentation with models and their algorithms and architecture. Section 3 and 4 discuss the workflow of the project and the necessary libraries and software perquisites for the project to successfully execute and the final section will help replicate the work done in this project.

A. *Data Pre-Processing and Analysis*

The abalone dataset contains 9 columns, wherein one variable is categorical and the rest of them are continuous variables. Table I, contains the details of all the 9 features including the units in which they are measured. The data was split into 2 parts for training and testing respectively, 30% was used in testing and 70% was allocated to training purposes.

TABLE I FEATURE DESCRIPTION

Attribute	Type	Values	Units
Sex	Categorial	0,1,2	
Length	Continuous	0.075 - .815	mm
Diameter	Continuous	0.055 – 0.65	mm
Height	Continuous	0.0 – 1.13	mm
Whole Weight	Continuous	0.002-2.8255	grams
Shucked Weight	Continuous	0.0010-1.488	grams
Viscera Weight	Continuous	0.0005–0.76	grams
Shell Weight	Continuous	0.0015-1.005	grams
Rings	Integer	1.0 – 29.0	Years

As there was one categorical variable in the dataset, I converted that variable to continuous values as it is much easier to process. The three categories were “M” for male, “F” for female and “I” for infant I converted them to 0 for male, 1 for female and 2 for infants. Furthermore, I converted the given problem into classification problem rather than regression problem by dividing the target variable into 4 categories. Abalones aged between 0-7 were labelled as “class 1”, abalones between 7-10 were labelled as “class 2”, instances aged between 10-15 were sent to “class 3” and the abalones between 15-29 were sent to "class 4”. Scaling of data is not performed as ensemble models as well as decision tree is not affected with normalization of the input features.

From Fig.1. we can infer that majority of attributes in our dataset do not demonstrate a Gaussian (normal) distribution. Among them, the 'height' attribute appears to exhibit a relatively normal distribution. However, both 'diameter' and 'length' attributes showcase a slight right-skewed distribution, indicating a prevalence of values leaning towards higher measurements. Conversely, the remaining attributes exhibit left-skewed distributions, suggesting a preponderance of values towards the lower end of the range.

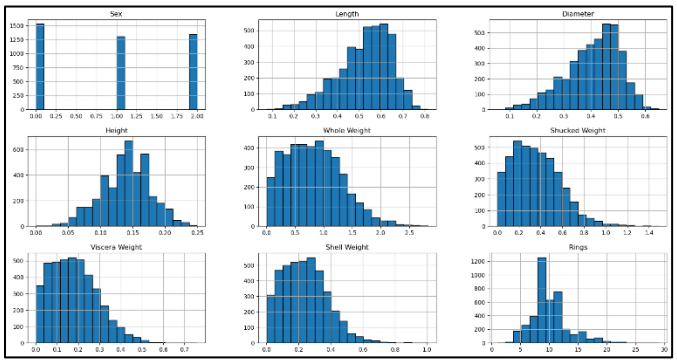

Fig.1 Feature Distribution

Fig.2 is a heatmap of correlation matrix which gives insight about multivariate relationships. The last row displays each numerical characteristics linear correlation with the target variable. We can clearly observe that there is no strong correlation between the target variable “ring” and the independent variables as most of the values lie between 0.5-0.65, “Shucked Weight” has the lowest correlation with the ring variable of 0.42 hence we will drop this in linear regression and will not consider it.

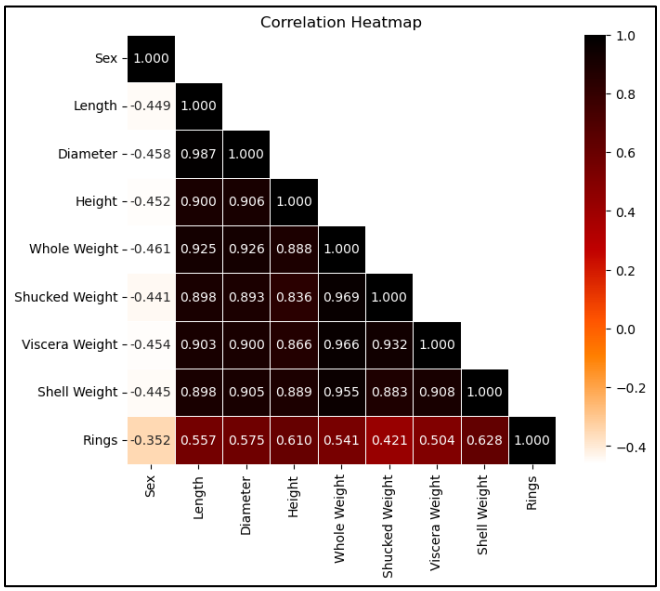

Fig.2 Correlation Heatmap

The probability plot (Chambers et al., 1983) is a graphical technique for assessing whether or not a data set follows a given distribution such as the Normal/Gaussian or Bernoulli [10]. For a dataset to follow a Gaussian distribution all the markers should form an approximate straight line on the red line and any significant deviation indicate that the distribution is skewed. Fig.3 indicates more positive skewedness.

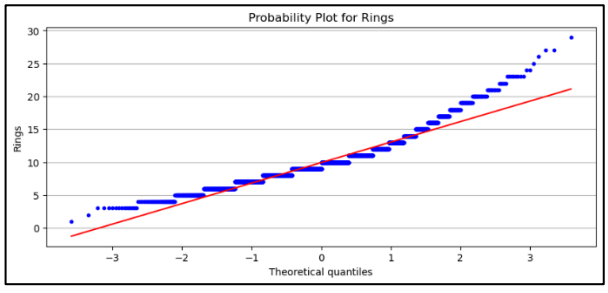

Fig.3 Q-Q plot of Rings

B. Model Overview

B.1 Decision Tree

It is widely employed algorithm in supervised machine learning, it serves purposes in both classification and regression tasks. It builds a tree like structure by recursively dividing the dataset based on the most significant attribute at each node, with a goal of maximizing information gain or minimizing the impurity, creating branches that lead to leaf nodes which represents the final prediction or outcomes [11]. To decide the feature based on which the data is to be split we use Information Gained, the feature having highest information gained will be used to split the data.

Entropy of a set S:

$$H(S) = -P_+ \log_2(P_+) - P_- \log_2(P_-)$$

Gain of a set S for an attribute f:

$$Gain(S, f) = H(S) - \sum_{v \in \text{val}} \left(\frac{|S_v|}{|S|} \right) \cdot H(S_v)$$

To overcome the issue of overfitting we can implement pruning. Pruning cuts, the unnecessary nodes which will reduce the complexity of the model which will reduce the chances of model overfitting [11]. There are two types of pruning Post-pruning, and Pre-pruning. Fig.4. displays post-pruning model in which we construct the decision tree first and then prune based on the results we get. From the example we can see that there is no need to further divide data after C5 as 90% of the data suggests the output is Y and only 10% of the data suggests it is N, hence we prune the tree and will not further divide.

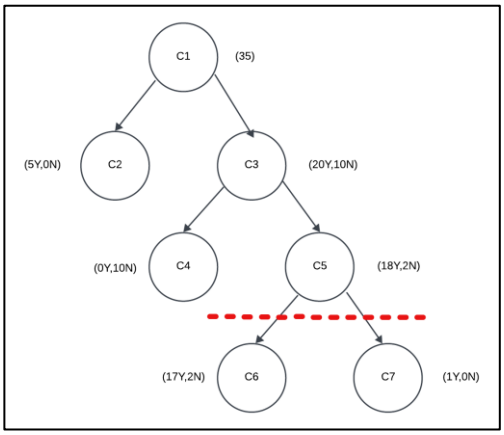

Fig.4 Post-pruning of Decision Tree

The second type of pruning is pre-pruning, in this case we first choose all the parameters with the help of GridSearchCV and then construct the tree with that "criterion", "max_depth", "splitter", "min_sample_leaf" and "min_sample_split". Pre-pruning is much faster in comparison to post-pruning and is less resource intensive due to these reasons I have implemented pre-pruning in this project.

B.2 Random Forest Classifier

Random forests consist of tree predictors, where each tree relies on values from a randomly sampled vector. The sampling is independent and follows the same distribution for all trees within the forest. As the number of trees in the forest increases, the generalization error for random forests asymptotically converges to a limit [12]. It is a bagging classification technique. Random forest performs row sampling as well as feature sampling with replacement on the given dataset, this means that there might be common sets of records and features present in two decision trees. Due to this random selection of features and records the variance reduces which resolves the issue of overfitting. Overfitting problems can be observed in decision trees but because of multiple decision trees in random forest and aggregation of their results we resolve this problem. Table 2 shows the parameters and their selected values used in the final implementation [13].

TABLE II. PARAMETERS USED IN RANDOM FOREST

Parameters	Description	Selected Value
Criterion	Quality of the split is assessed using these functions.	log-loss
n_estimators	It is the number of trees in the forest.	125
max_depth	The maximum depth of the tree.	6
ccp_alpha	It is the parameter for Minimal Cost-Complexity Pruning. The chosen subtree will be the one with the largest complexity below ccp_alpha.	0.015

B.3 Gradient Boosting Algorithm (GBM)

Boosting: It is an ensemble technique in which we combine multiple models. When combining these are individually weak learners but when combined they become strong learners and reduce the training error. In boosting a random sample of data is selected and trained sequentially, in this each model tires to compensate for the error made by the previous model. In the end the output which is given has low training and testing error [14].

Gradient Boosting:

Algorithm 2 Gradient Boosting Algorithm

Step 1: Build a base model to predict training data by taking the average of the target column.
Step 2: Calculate pseudo residuals ($observedvalue - predictedvalue$).
Step 3: Build a model on pseudo residuals, generating error values as predictions.
Step 4: Find output values for each leaf of the decision tree (average for regression, majority vote for classification).
Step 5: Update the prediction of the previous model:

$$NewPrediction = PreviousPrediction + LearningRate \times Treemadeonresiduals$$

Source of this algorithm was research paper written by Hastie, T Et.el [15]

For implementation of this algorithm default parameters were used, Table III shows the values and parameters:

TABLE III. PARAMETERS USED IN GRADIENT BOOSTING

Parameter	Description	Selected Value
Loss	It is the loss function to be optimized.	log_loss
Learning rate	It shrinks the contribution of each tree by factor of learning_rate.	0.1
N_estimators	The number of boosting stages to be performed.	100

B.4 Extreme Gradient Boosting (XGBoost)

XGBoost adheres to the principles of gradient boosting, but it distinguishes itself through variations in modeling specifics. It employs a more stringent model regularization approach to mitigate overfitting, resulting in enhanced performance [16].
It undergoes optimization via parallel processing, tree-pruning, addressing missing values, and regularization to prevent overfitting or bias. XGBoost improves upon GBM by optimizing both systems and algorithmic aspects. As depicted in Fig.5, XGBoost represents an advancement over GBM [17].

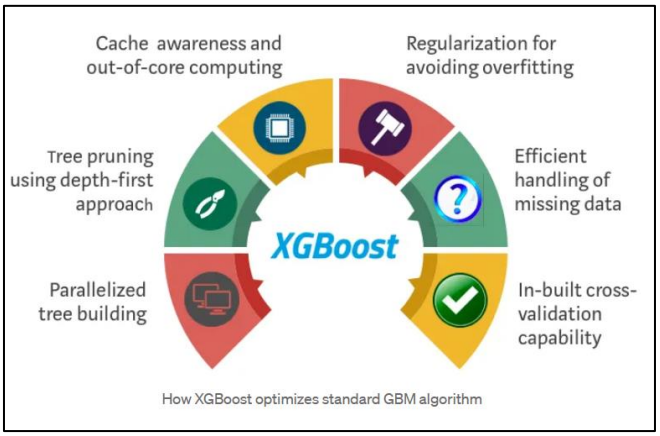

Fig.5 How XGB optimizes GBM algorithm [16].

Default parameters were used in implementation of XGBoost classification algorithm.

B.5 Neural Network

Neural Network consists of multiple components which includes input layer, hidden layers, output layers, weights, bias, and activation function. During the feedforward part the input layer transmits the data through hidden layers (acts as processing unit) onto the output layer. Using a selected loss function, the network then computes the error between the expected and actual outputs during backpropagation. The error is then propagated backward through the network, with optimization techniques such as gradient descent being used to adjust the weights and biases in the direction that minimizes the error.
For implementation in this project, I have used ADAM optimizer with 3 hidden layers using ReLu as activation function and learning_rate set to 0.1.

TABLE IV. PARAMETERS USED NEURAL NETWORKS

Parameters	**Description**	**Values**
learning_rate	This updates the weights	0.01
solver	Used for optimizing weights.	ADAM
max_iter	Maximum number of iterations.	500

B.6 Random Forest Regressor

Random Forest Regressor works on the same principle as the random forest classifier does, but this algorithm is used in prediction of continuous values. Like classifier there are multiple decision trees which predict various numerical values as their output but rather than taking a majority vote as we do in random forest classifier, in regressor we take the mean value of all the outputs as the predicted value [18].

In random forest D', D'', D''', D'''' will always be less than the size of original dataset. From Fig.6 we can see that there are 4 decision trees, we have just considered a simple example. The output of all the trees is 1.34,1.38,1.32 and 1.35 respectively. The random forest regressor output for this dataset will be following:

$$Output = \frac{1.34+1.38+1.32+1.35}{4} = 1.3475$$

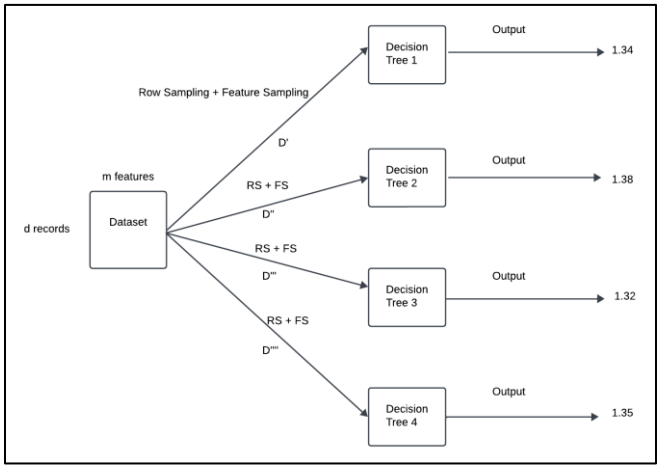

Fig 6. Workings of Random Forest Regressor

C. Workflow Diagram

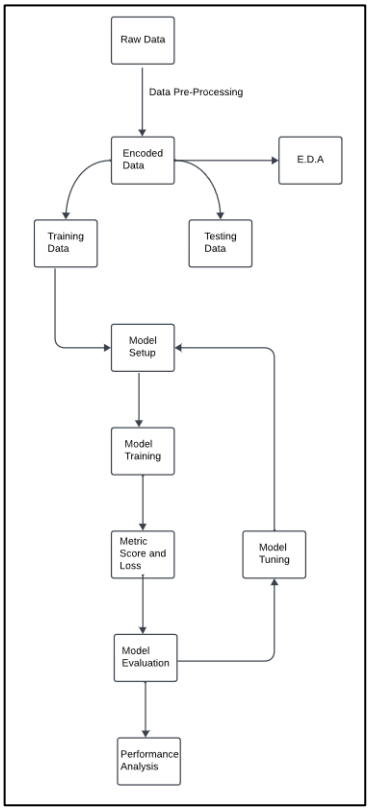

Fig.7 Experimental Workflow

The project begins with processing the data, then performing EDA and continues to model setup, model evaluation and finally performance analysis.

D. Software Suite

This project was implemented in Python programming language using Visual Studio as the editor.

Table V has all the libraries and packages that we require to implement this project.

TABLE V NECESSARY LIBRARIES

Libraries	Packages	Classes
pandas	*	*
numpy	*	*
sklearn	model_selection	train_test_split
		GridSearchCV
	tree	
	metrics	accuracy_score, classification_report, confusion_matrix
	impute	SimpleImputer
	ensemble	GradientBoostingRegressor, GradientBoostingClassifier, RandomForestClassifier
	preprocessing	LabelEncoder
	neural_network	MLPClassifier
seaborn	heatmap	*
scipy	stats	probplot
matplotlib	pyplot	*
xgboost	xgb	*

"*" indicates all the modules were imported.

IV. Model Development and Performance Analysis

Training dataset was used to train all the models, the pre-processed data was split into 70:30 ratio and assigned to training and testing respectively. The model assessment was done by measuring the performance of models by using various evaluation metrics.

A. Evaluation Metrics

For classification problem, I have used 2 evaluation metrics (1) Accuracy, (2) Cohen's Kappa coefficient, (3) F1-Score. F1-Score is a better evaluation metric for this dataset because we can observe class imbalance and F1-Score gives equal weightage to all the classes. Accuracy measures the overall correctness of the model's prediction and is given by [19]:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

The F1 score is calculated as the harmonic mean of precision and recall, taking into account both false positives and false negatives. [19].

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Matthews Correlation Coefficient (MCC) considers true positives, true negatives, false positives, and false negatives. It is very useful in case of imbalanced data, as in our case 4th class of abalones aged between 15-29 have very low number of records this metric can be applied for evaluation. The range extends from -1 to 1, where +1 signifies flawless prediction, 0 denotes no improvement over random prediction, and -1 indicates complete contradiction with the prediction [20]. MCC is preferred over Cohen's Kappa Coefficient because of its ability to class sensitivity and handling imbalance classes as it includes both false positive and false negative values [21].

$$MCC = \frac{(TP * TN - FP * FN)}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}}$$

For Regression problem, I have used 2 more evaluation metrics (1) Root Mean Squared Error (RMSE) and (2) Mean Average Error (MAE). RMSE calculates the residual (predicted - actual), computes the norm of each observation then takes the mean of all the observation and takes the square root of that mean.
MAE is calculated as the average magnitude of errors between the predicted and the actual values [24].

B. Parameter Tuning

1) Decision Tree

The following table describes the tuned parameters used for decision tree model in this project.

TABLE VI. TUNED PARAMETERS DECISION TREE

Parameters	Description	Value
Criterion	Quality of the split is assessed using these functions.	'gini'
Max_depth	The maximum depth of the tree.	4
Splitter	It is employed to select the split at each node.	'best'

Pre-pruning was performed on this model and following are the parameters post pruning:

TABLE. VII TUNED PARAMETERS POST PRUNING

Parameters	Description	Value
Criterion	Quality of the split is assessed using these functions.	'gini'
Max_depth	The maximum depth of the tree.	4
Splitter	It is employed to select the split at each node.	'best'

2) Random Forest Classifier

TABLE. VIII TUNED RANDOM FOREST CLASSIFIER

Parameters	Description	Value
N_estimators	The number of trees in the forest.	125
Criterion	Quality of the split is assessed using these functions.	'log_loss'
Ccp_alpha	It is the parameter for Minimal Cost-Complexity Pruning.	0.015
Max_depth	The maximum depth of the tree.	6

3) Neural Networks

TABLE. IX TUNED PARAMETERS NEURAL NETWORKS

Parameters	Description	Value
Activation	The function utilized for activating the hidden layer.	'ReLu'
Solver	The algorithm used for optimizing weight parameters.	'adam'
learning_rate	This parameter decided the weight updating.	'adaptive'
Maximum iterations	Maximum number of iterations.	500

Default parameters were used in implementation of Gradient boosting algorithm and Extreme Gradient Boosting algorithm.

4) Random Forest Regressor

The following table describes the tuned parameters of the random forest regressor, k-fold cross validation was performed with 5 folds on 10 candidates totaling 50 fits to obtain this result.

TABLE. X TUNED RANDOM FOREST REGRESSOR

Parameters	Description	Value
N_estimators	The number of trees in the forest.	200
Minimum Sample Split	The smallest number of samples needed to divide an internal node.	10
Minimum Sample Leaf	The minimum number of samples needed to form a leaf node.	2
Maximum Depth	The maximum depth of the tree.	50
Maximum Features	The count of features to contemplate while searching for the optimal split.	'sqrt'

V. Results and Analysis

All the models implemented in this project were trained on the training set and tested using the evaluation metrics with tuned hyperparameters.

A. Decision Tree

An accuracy of .58 was achieved, with f1-score (macro) 0.47 recorded and Matthew's coefficient of 0.36 was achieved which indicates that there was less agreement between the predicted and the observed values of the target class.

After performing pre-pruning technique which is used to reduce the overfitting problem these scores were improved as observed from the table below.

TABLE. XI COMPARISION OF PRUNED AND SIMPLE DT

Model	Accuracy	F1-Score	Matthew's Coefficient
Simple Decision Tree	0.58	0.47	0.36
Pre-Pruned Decision Tree	0.60	0.52	0.39

B. Random Forest Classifier

The following figure describes the accuracy of random forest classifier by depth. Initially as the depth increases the accuracy also increases as the model gets complex, but after a certain depth the accuracy starts to reduce due to overfitting problem. It can be observed clearly as the accuracy increases to a depth of 6 and then starts to fluctuate.

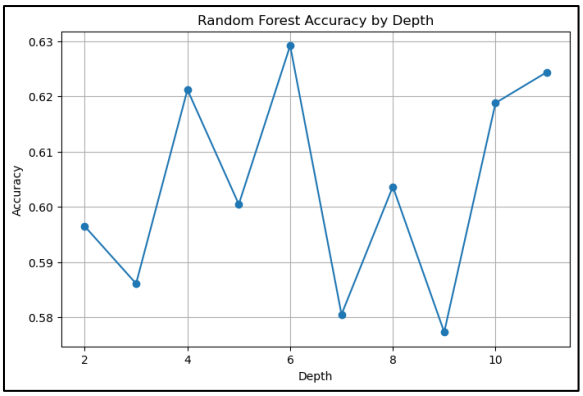

Fig. 8 Accuracy of Random Forest by Depth

Post hyperparameter tuning, the highest accuracy of 0.63 was achieved with the highest Matthew's Coefficient being 0.42. Ten experiments were performed using the tuned parameters, the mean accuracy of all the runs came out to be 0.60.

C. *Gradient Boosting Algorithm*

Fig. 7 describes the confusion matrix of gradient boosting algorithm, best accuracy was achieved while prediction of class-2 (10-15) as it predicts 471 records correctly, we can also observe poor accuracy in case of class 4 because of a smaller number of records present of that class in the dataset.

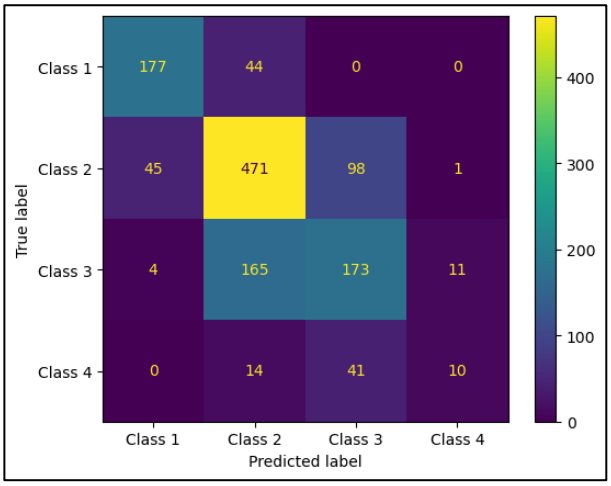

Fig. 9 Confusion Matrix of Gradient Boosting Algorithm

10 experiments were conducted using the tuned hyperparameters and a mean accuracy of 0.61 was achieved, 0.46 Matthews coefficient was achieved with mean 95% confidence interval of the accuracy being 0.59-0.61.

D. *Extreme Gradient Boosting*

As XGBoost algorithm is an optimized version of gradient boosting algorithm, we can observe improvement in the evaluation metrics. I was able to achieve a mean accuracy of 0.63 over 10 experiment runs. Matthew's score of 0.48 was achieved and F1-score of 0.56 was recorded. The mean 95% confidence interval of 0.62-0.63 was observed over 10 experiment runs.

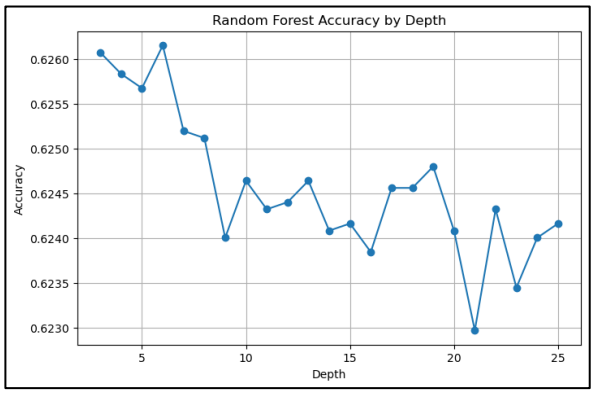

Fig.10 Accuracy by depth of XGBoost Algorithm

From Fig.8 we can confirm the fact that the highest accuracy achieved was 0.63 and it was achieved with a depth of 6 levels.

E. *Neural Network*

Fig. 10 displays the confusion matrix of neural network with best parameters. From this figure it can be inferred that the highest accuracy was achieved in predicting class 2 as 468 instances were predicted correctly.

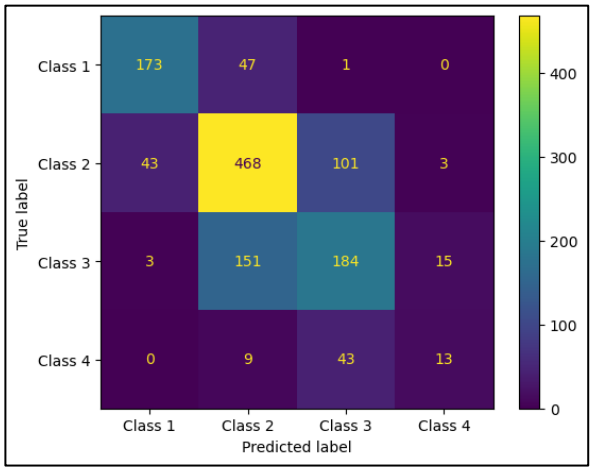

Fig.11 Confusion Matrix of Neural Networks

This model performed the best among all the models achieving mean accuracy of 0.64 and a mean F1-score of 0.565. The mean 95% confidence interval over 10 observations was 0.63-0.65 and Matthew's Coefficient of 0.475 was achieved.

F. *Random Forest Regressor*

I performed k-fold cross validation on the dataset to achieve 0.562 R-squared score and was able to achieve a Mean Squared error score of 0.55. In comparison to the previous assessment this model was able to achieve better R-squared value than neural network used for regression without any tuning and was just second to tuned multi-layered perceptron model. 0.562 R-Squared value means that this model was able to explain 56.2% of the variance in the dependent variable that is explained by the independent variables in the regression model.

G. Comparision of Models

Fig.11 shows us the comparison of accuracies of 3 ensemble techniques Random Forest Tree, Gradient boosting and XGBoost. We can observe that over 10 experiments XGBoost performs the best in comparison to Gradient boost model as it is modified version of GB and Random Forest classifier performs the worst in comparison to these two algorithms.

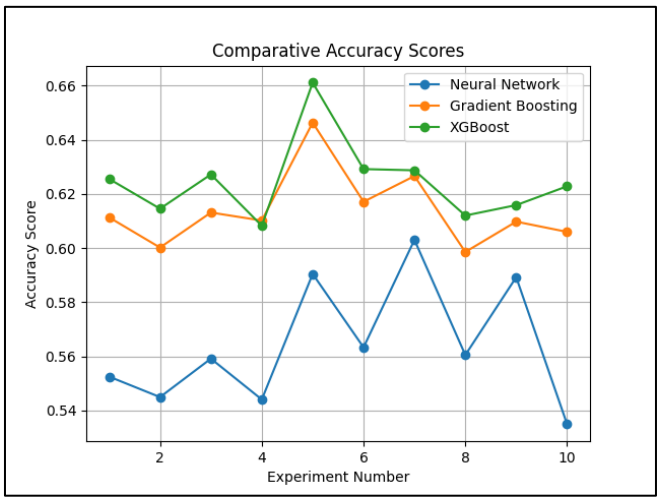

Fig 12. Comparison of Accuracies of 3 models.

TABLE. XII RESULTS

Model	Accuracy	F1-Score	Matthew's Coefficient	95% confidence interval of accuracy
Decision Tree	0.607	0.562	0.389	0.584 - 0.63
Decision Tree with Pruning	0.61	0.51	0.41	-
Random Forest Classifier	0.60	0.455	0.378	0.588 - 0.612
Gradient Boosting	0.614	0.50	0.465	0.605- 0.618
XGBoost	0.625	0.55	0.466	0.615 – 0.634
Neural Network	0.64	0.565	0.475	-

- The Neural Network consistently performs well across all metrics, with the highest accuracy, F1-Score, and Matthew's Coefficient.
- XGBoost also performs competitively, especially in terms of accuracy and F1-Score.
- Decision Tree with Pruning shows improvements over the regular Decision Tree, but its performance is still moderate compared to other models.
- Gradient Boosting performs well, particularly in terms of Matthew's Coefficient.
- Random Forest Classifier has a moderate performance but falls behind the Neural Network, XGBoost, and Gradient Boosting.

VI. Conclusion

In this study, we conducted a thorough comparison of five models—Decision Tree, Decision Tree with Pruning, Random Forest Classifier, Gradient Boosting, and XGBoost—employing three comprehensive evaluation metrics: Accuracy, F1-Score, and Matthew's Coefficient. The major contributions of our research include the detailed performance analysis of these models across multiple dimensions and the utilization of diverse evaluation metrics for a nuanced understanding of their strengths and weaknesses. Our findings highlight Neural Network as the top performer, closely followed by XGBoost. For future research, addressing class imbalance, hyperparameter tuning for XGBoost and Gradient Boosting, exploration of advanced neural network architectures such as CNNs and RNNs, and investigating ensemble approaches stand out as promising avenues to further enhance the classification performance on the dataset.

VII. References

1) L. Chen and J. Ryan, "Abalone in Diasporic Chinese Culture: The Transformation of Biocultural Traditions through Engagement with the Western Australian Environment," *Heritage*, vol. 1, no. 1, pp. 122–141, Jul. 2018, doi: https://doi.org/10.3390/heritage1010009.

2) M. Hossain, M. Niaz, and M. Chowdhury, "M P RA Munich Personal RePEc Archive Econometric Ways to Estimate the Age and Price of Abalone Econometric Ways to Estimate the Age and Price of Abalone," 2019. Available: https://core.ac.uk/download/pdf/214010482.pdf

3) Tarbath, David & Officer, Rick. (2003). SIZE LIMITS AND YIELD FOR BLACKLIP ABALONE IN NORTHERN TASMANIA.

4) Ho, T. K. (1995, August). Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 1, pp. 278-282). *[IEEE]*

5) "UCI Machine Learning Repository," archive.ics.uci.edu. http://archive.ics.uci.edu/dataset/1/abalone.

6) Learning Decision Trees – [Harryt].

7) M. F. Misman *et al.*, "Prediction of Abalone Age Using Regression-Based Neural Network," *IEEE Xplore*, Sep. 01, 2019. https://ieeexplore.ieee.org/document/8970983

8)Aalayya. Predicting age of Abalone using Neural Networks [Link].

9) Kingma, D.P. (2014) Adam: A method for stochastic optimization. https://arxiv.org/abs/1412.6980.

10) A. Bataa, "Prediction of approach time to significant points in aircraft trajectories." [Online]. Available: https://dspace.cvut.cz/bitstream/handle/10467/103934/F6-BP-2022-Bataa-Anu-BP%20-%20Anu%20Bataa%20-%20final.pdf?sequence=-1&isAllowed=y

10) Chambers, J. & Cleveland, William & Kleiner, B. & Tukey, P. (2012). Graphical Methods for Data Analysis (vol 17, pg 180, 1983). Journal of Sleep Research. 21. 484-484. 10.1111/j.1365-2869.2011.00992.x.

11) Patel N, Upadhyay S. Study of various decision tree pruning methods with their empirical comparison in WEKA. Int J Comp Appl. 60(12):20–25. [Google Scholar]

12) Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32: https://www.cise.ufl.edu/~anand/fa11/Breiman_Random_Forests.pdf

13) https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

14) Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. Frontiers in neurorobotics, 7, 21: https://www.frontiersin.org/articles/10.3389/fnbot.2013.00021/full

15) Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

16) Lecture Notes Week – 8: https://edstem.org/au/courses/13855/lessons/41914/slides/289365

17) Morde, V. (2021) 'XGBOOST algorithm: Long may she reign! - towards data science,' Medium, 9 December. https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d.

18) Beheshti, N. (2022) 'Random Forest regression - towards data science,' Medium, 5 March. https://towardsdatascience.com/random-forest-regression-5f605132d19d.

19) Powers, D. M. (2020). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. arXiv preprint arXiv:2010.16061. https://arxiv.org/pdf/2010.16061.pdf

20) Wikipedia contributors (2023) Phi coefficient. https://en.wikipedia.org/wiki/Phi_coefficient.

21) Delgado R, Tibau XA. Why Cohen's Kappa should be avoided as performance measure in classification. PLoS One. 2019 Sep 26;14(9):e0222916. doi: 10.1371/journal.pone.0222916. PMID: 31557204; PMCID: PMC6762152.

22) Pascual, C. (2023) Tutorial: Understanding regression error metrics in Python. https://www.dataquest.io/blog/understanding-regression-error-metrics/.