Decision Tree, Ensemble Learning and Simple Neural Network

*Comparison of three different models(Decision Tree, Ensemble Learning and Simple Neural Network) with different improving algorithm

1st Yuting Fang
MATH3856
University of New South Wales
Sydney, Australia
z5518340@ad.unsw.edu.au

1st Bingchen Xie*
MATH5836
University of New South Wales
Sydney, Australia
z5610033@ad.unsw.edu.au

1st Runyu Han
MATH3856
University of New South Wales
Sydney, Australia
z5387147@ad.unsw.edu.au

***Abstract*—The objective of this study is to evaluate and compare the performance of three machine learning models, namely decision trees, ensemble learning, and simple neural networks, with a particular focus on their ability to accurately classify abalone age (Class 1: 0 - 7 years; Class 2: 8- 10 years; Class 3: 11 - 15 years; Class 4: Greater than 15 years) using physical characteristics. Furthermore, two datasets of varying sizes were selected to investigate the models' performance on large and small datasets. The optimal method was identified through an analysis of the various hyperparameter configurations and dataset sizes. Methodologically, the study employed pre- and post-pruning techniques for decision trees, optimization of random forests and boosting models within an ensemble learning framework, and regularization and dropout techniques for neural network models. The models were tested on three datasets: abalone, a smaller contraceptive method choice (CMC) set, and a larger adult income dataset. The results demonstrated that the XGBoost model exhibited the highest level of accuracy, with a test accuracy of 0.632, on the abalone dataset. In the case of the smaller dataset (CMC set), the gradient boosting method demonstrated superior performance, with a test accuracy of 0.561. Conversely, the random forest method exhibited notable efficacy on the larger dataset (adult dataset), with a test accuracy of 0.923 and an F1 score of 0.9214. These findings suggest that boosting models may be more suited to smaller datasets, while random forests may offer enhanced performance on larger datasets.**

***Index Terms*—Decision Tree, Random Forest, Boosting, Pre-pruning, Post-pruning, Simple Neural Network, Abalone, regularization, Dropout, Large dataset, Small Dataset, hyperparameters**

I. Introduction

A. Background

Decision Tree learning algorithm is one of the most widely used machine-learning methods for classification and regression problems, constructed by "a divide-and-conquer approach" [1]. Since it is intuitive and simple to implement, it has been used to assess financial credit scores [2], analyze users' preferences in marketing [3], and perform more analogous tasks in other areas. The random forest is constructed by multiple decision trees, which can better handle overfitting and noisy problems than the decision tree method. Since the random forest is flexible and powerful, it is generally used in more places, such as seeking groundwater [4] and building environment problems [5]. Furthermore, the simple neural network was the main method used for regression and classification tasks by imitating the human learning process, especially in building the basics of many modern techniques [6]. Meanwhile, boosting is becoming increasingly critical in machine learning. This method relies on a set of weak learners, utilizing their errors to iteratively update weights and fit a strong learner, generally having high performance and excellent ability to generalize [7].

From the literature review, we find that Machine learning faces some issues when trying to find the best model. Overfitting represents a significant challenge in supervised machine learning, which hinders the ability to accurately generalize models to both observed and unseen data, particularly when evaluated on a testing set [8]. Meanwhile, insufficient sample size often leads to problems when doing both Neural Networks and Decision Tree [18] [10] Furthermore, the issue of hyperparameter optimization is a topic of considerable debate, as it has a direct bearing on the performance of a model [11].

Moreover, as the global population expands, the international market for abalone, a renowned and highly sought-after seafood delicacy, has witnessed considerable growth in recent years, with the abalone industry becoming a significant contributor to the global economy [12]. The age of the abalone is a significant determinant of its price, as it is the primary factor used to ascertain its value [13]. The formation of the rings is the result of the incremental growth of the abalone's inner shell, occurring at a rate of one ring per year [13]. A review of the literature shows that the artificial intelligence approaches currently available for predicting the age of abalone are still inadequate. [13].

B. Motivations

Based on the above background, loads of issues remain unresolved. The application of machine learning methods inevitably results in the occurrence of overfitting and insufficient data volume. Furthermore, the process of hyperparameter optimization presents a significant challenge. Simultaneously, ar-

tificial intelligence approaches currently available for abalone age prediction are still inadequate [13] There are numerous issues that require further investigation and resolution.

C. Contribution

This paper will discuss three methods (Decision Tree, Ensemble Learning, and Simple Neural Network), which are used to classify abalone age into four classes (Class 1: 0 - 7 years; Class 2: 8- 10 years; Class 3: 11 - 15 years; Class 4: Greater than 15 years) based on eight physical features and will identify the optimal two models for this dataset. Moreover, the article will present a methodology for selecting the optimal classification method and applying it to two further sets, including small and large instances respectively, to facilitate a comparison of the influence of the size of the datasets. The details are as follows:

- The original three data sets are all from UC Irvine, used for predicting the ring age of abalone, choosing the best contraceptive method, and telling the income of adults.
  - Abalone set is used for predicting the ring age of abalone, with no missing values. Inside this data, there are 4177 instances of eight features. All the variables have the continuous type, except the "Sex" data which are categorical to "M(Male)", "F(Female)" and "I(Infant)" and the "Rings" data are integers.
  - CMC set is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey, with 10 variables and 1473 instances, and no missing value, aiming at classifying three different types of contraceptive methods. The other nine variables are features, where "wife age" and "num children" are continuous, "wife edu", "husband edu", "husband occupation", and "standard of living index" are categorical, "wife religion", "wife working", "media exposure" are binary.
  - Adult set is a complex dataset, with some missing values, for predicting whether people's income is over 50,000 dollars a year, containing 15 variables. Among these, The "age", "fnlwgt", "education-num", "capital-gain", "capital-loss", "hours-per-week" are continuous features. The "workclass", "education", "marital-status", "occupation", "relationship", "race", "native-country", "sex" are categorical features. Except that, all complete variables contain 60000 instances.
- Trying to find the best two models among Decision trees, Simple Neural Networks, and Ensemble Learning on the abalone set when classification and comparing the different methods used to improve the model. Then, we will compare the following combinations.
  - For the Decision Tree, we examine different groups of hyperparameters to find the best one (pre-pruning) and post-pruning to improve the model further.
  - For the Ensemble Learning, we need to compare the performance of Random Forest, Gradient Boosting, and XGBoost.
  - For Simple Neural Network, we compare the two different optimizers, SGD and Adam, choosing the best one. Then, compare the consequences after using two methods, L2 regularization, and Dropout to prevent overfitting.

  After applying the above method to the abalone set, we can pick the two best models, and then use the models to the CMC set to test the performance in another small size of data. Then, repeat the overall process to find the best classifier for Adult sets and compare those results, to find which of the models will perform better in large-size datasets and which will do better when applied to a small dataset.

D. Overview

In this article, we use four different sections to present our research. In section II, we mention all the settings in our experiments and the mathematical explanation of the used methods. In section III, the results of our experiments are reported. In section IV, we include the summary and the limitations of our experiments. In section V, we emphasize the contribution and discuss the further improvement of our methodology.

II. Methodology

A. Data Processing

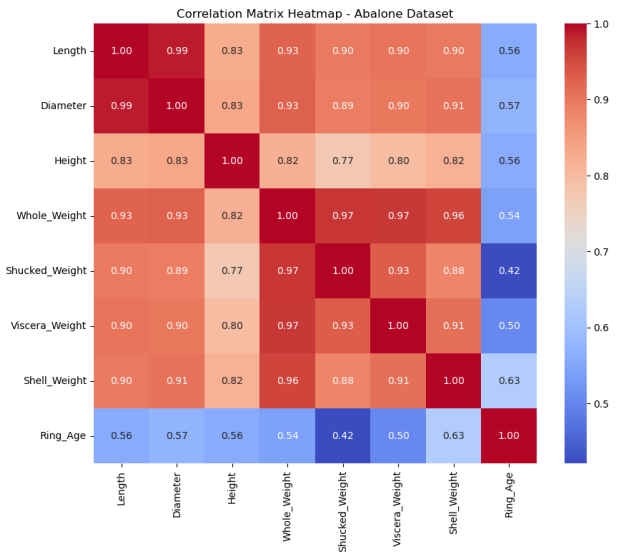


Fig. 1: Correlation Heatmap of Features of Abalone

In our three datasets, there are many categorical and binary variables. Our group would use one-hot to process these variables. Each Label in the category would be considered a new variable and allotted 0 or 1 based on whether the sample belongs to this category. Moreover, all datasets used for training the neural network model would be scaled to make better model performance. Since the Adult dataset has some missing values. Our group would use medians to replace the continuous missing values and majorities to replace the

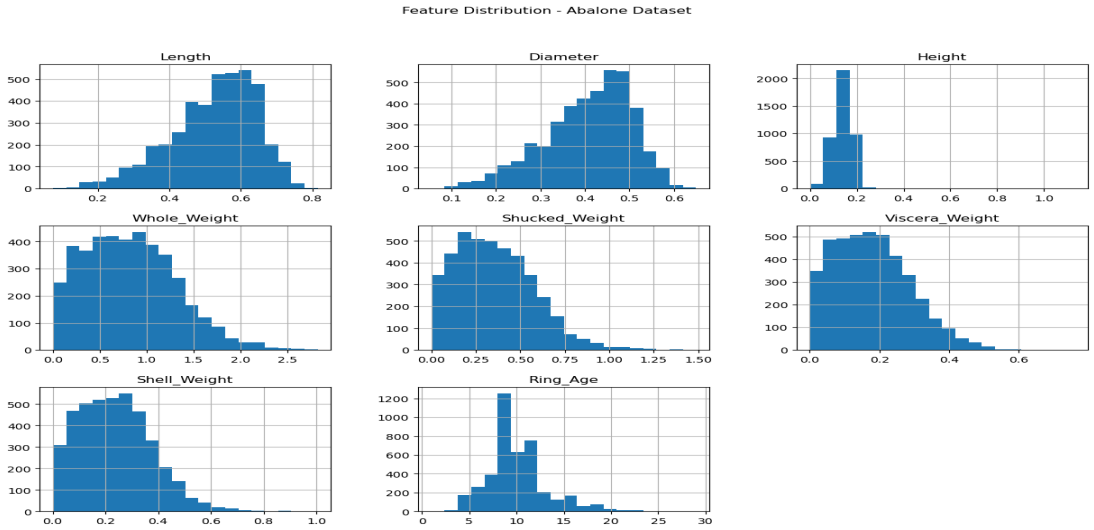


Fig. 2: Abalone Histograms for each Feature

missing values in discrete variables. Furthermore, “fnlwgt” is the weight of each sample in the adult set, which we decided not to use in our model. Meanwhile, ”education” is a different description of ”education-num”, so we delete it when we apply models.

- Abalone Set Figure 1 shows the correlation between the variables in the abalone dataset. According to the map, all features are highly correlated with other features, especially the “Length” and the “Diameter” (0.99). The target “Rings” has the highest correlation values with “Shell weight” (0.63) and “Diameter” (0.57). Figure

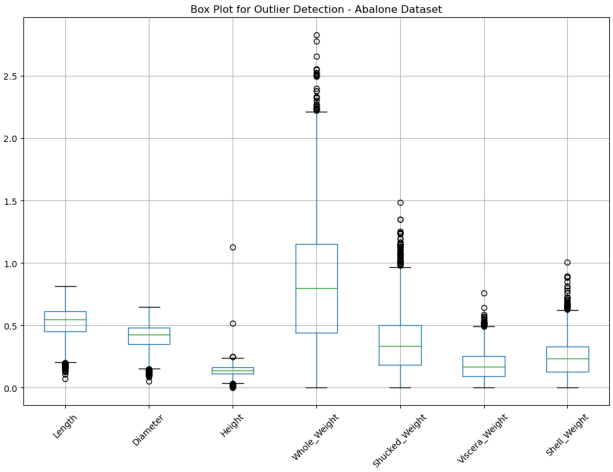


Fig. 3: Boxplot of Feature in Abalone

2 shows the histograms of variables in the abalone dataset. By this figure, these variables in the abalone dataset do not tend to have a normal distribution. The histograms of the “Whole weight”, “Shucked weight”, “Viscera weight”, “Shell weight” are skewed to the left, additionally “Length” and “Diameter” are skewed to the right. The target “Rings” tend to have two peaks, but other variables only have one. Features “Shell weight” and “Shucked weight” tend to have long tail phenomenon. Most ”Rings” data concentrate on the middle area, which leads to an imbalanced data situation in the classification task, especially for multiple classes. Moreover, most of the data are focused, but values bigger than 0.8 in “Shell weight” and bigger than 1.25 in “Shucked weight” might be outliers.

Figure 3 is the boxplot of abalone data. Most data of variables are concentrated except “Whole weight” data. Two points of the variable “Height” higher than 0.5 could be outliers.

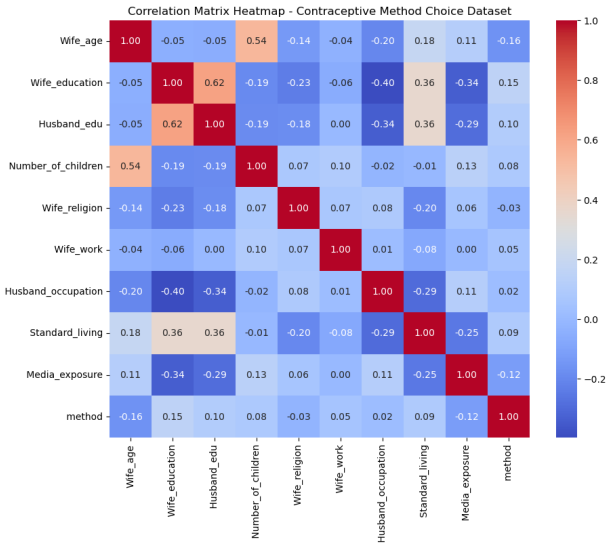


Fig. 4: Correlation Heatmap of Features in CMC Set

- CMC Set Figure 4 shows the target “contraceptive method” has a low correlation with all features. Most

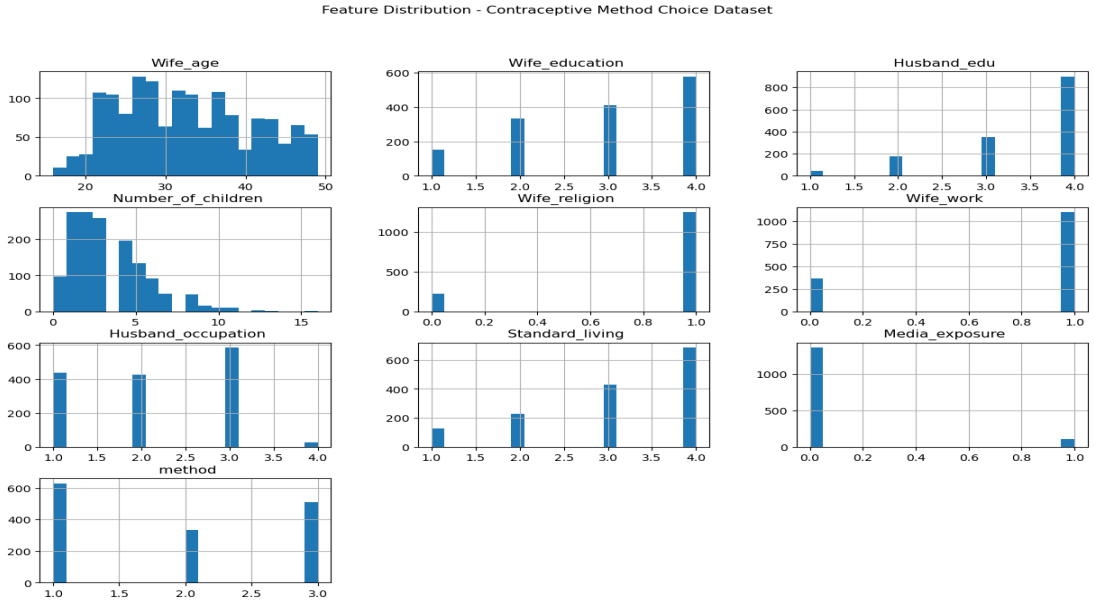


Fig. 5: Histograms of features in CMC set

of the correlation numbers are lower than 0.20. Inside these features, "wife age", "wife religion" and "media exposure" are negatively correlated with "contraceptive method" while other features are positively correlated. As for the correlation of features, only "husband edu" and "wife edu" have a comparatively high correlation, which is 0.62. Figure 5 is the histogram of the CMC dataset.

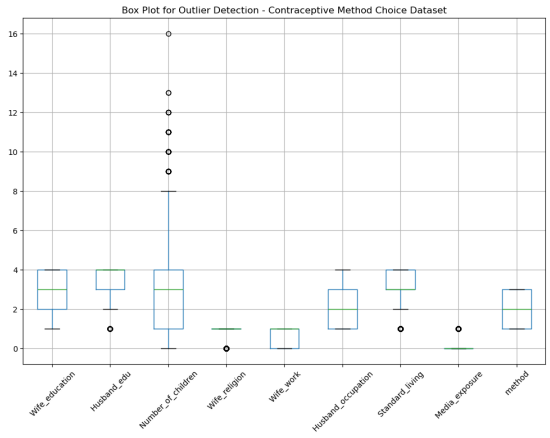


Fig. 6: Boxplot of features in CMC Set

The data on "wife age" is concentrated. Although it has multiple peaks, samples still exist in each age group. However, the data of "num children" is not. There are two ranges of "num children" having no value. Moreover, numbers bigger than 15 in "num children" could be considered outliers. For binary variables, "wife religion" and "media exposure" both exist in a class that is extremely larger than another one in sample numbers. Categorical variables tend to have a balanced distribution for each category. However, the categories "1" in "husband edu" and "4" in "husband occupation" are very low in sample numbers.

Figure 6 shows the distributions of "wife edu", "husband edu" and "standard living" might be skewed to the right. Outliers majorly exist in the variable "num children". Most of the data in variables "wife religion" and "media exposure" concentrate on one point, which is their median.

- Adult Set

  The original dataset is large and complex, making all the features visualized if not convenient. Hence, we only chose six feathers that are easy to show, and the graphs are as follows. Figure 7 shows that the correlations between all variables in the adult dataset are low.

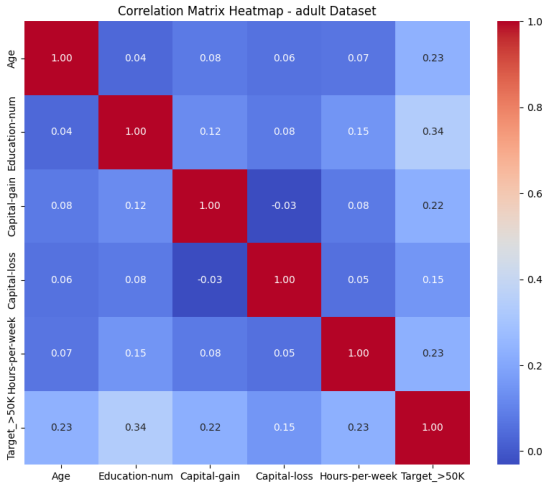


Fig. 7: Correlation Heatmap of some Features in Adult Set

Variables “education-num” and “Target” hold the highest correlation value (0.34), which means there is even no linear relation between those feathers. Hence, this dataset is complex datasets

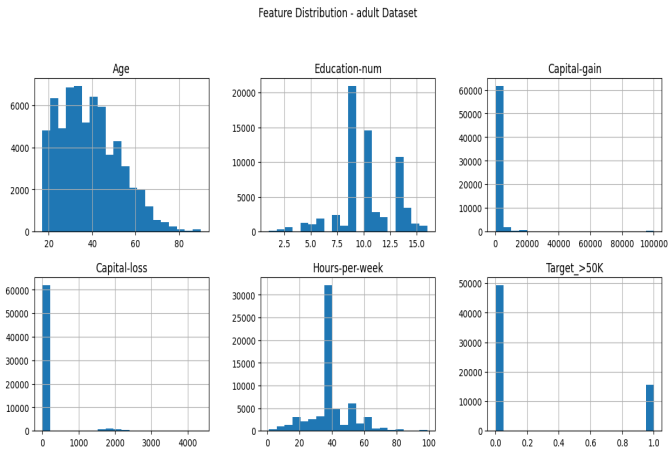


Fig. 8: Histograms of some features in Adult set

Figure 8 shows the distribution of all continuous variables. The “age” skew to the left, with a slight long tail phenomenon. Moreover, almost all “capital-gain” and “capital-loss” data concentrate on point 0. The data which is not in 0 might be considered outliers. Except that, “education-num” also has no data in some range. Furthermore, the ”Target” shows an obvious inequality between larger than 50k and smaller than 50k. By the

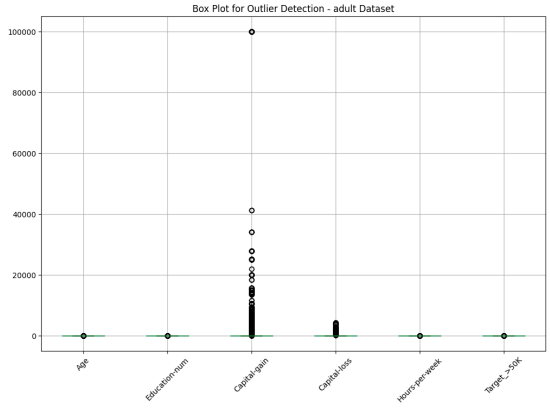


Fig. 9: Boxplot of some features in Adult Set

figure 9, data of all variables in the adult dataset are distribution concentration. A few outliers exist in features “capital-gain” and “capital-loss”.

B. Decision Tree

Decision tree is one of the most powerful research methods in artificial intelligence, which is commonly used when doing classification [14]. The fundamental concept underlying a decision tree is the construction of a model that makes decisions by partitioning the data into subsets based on the values of input features. The decision tree operates in a hierarchical, tree-like structure, wherein each internal node represents a decision regarding a feature, each branch represents the outcome of that decision, and each leaf node represents a final decision or prediction.

Information gain is the measure of how much information a feature gives about the classes, and the main goal of the decision tree algorithm is to maximize information gain. To process the method, we first need to choose the feature that has the greatest Information Gain:

$$\text{Gain}(S, A) \equiv \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \quad (1)$$

as the Root Note of the tree, where

$$\text{Entropy}(S) = -\sum_i p_i \log_2 p_i \quad (2)$$

and the S is the given dataset, p_i is the proportion of positive/negative examples in S. For the following leaves, we choose the feature in each note depending on the amount of Information Gain.

Nevertheless, Gini Gain

$$\text{Gini} = 1 - \sum_{i=1}^{n} (p_i)^2 \quad (3)$$

is another measurement for choosing notes, the CART Decision Tree algorithm will choose the attribute with the lowest Gini Gain as the Root Node. Also, we choose the following notes for each level of the tree depending on the Gini Gain, the lower the Gini Gain the more priority of a feature be a note in the Decision tree.

C. Post-pruning

Post-pruning, also referred to as ”pruning” or ”tree pruning,” is a technique employed to streamline a decision tree and mitigate overfitting by eliminating sections of the tree that are not essential for accurate prediction.

The initial step is to expand the decision tree to its maximum depth, divide the nodes, and then assess the effect of removing each node or branch using a cost-complexity measure:

$$\text{Cost}(T) = \text{Error}(T) + \alpha \times \text{Number of Leaves}(T), \quad (4)$$

where Error(T) is The error rate of the tree T on the training data, and α is a hyperparameter that controls the trade-off between the tree’s complexity and its accuracy. Higher values of α encourage more pruning. The goal is to minimize the cost complexity.

D. Random Forest

When we ensemble loads of Decision trees we got the method of random forest. To learn it, we need to do the sampling with replacement to the original dataset, then use the new dataset to establish a new Decision Tree, repeating the process until we get the highest accuracy on the test set, in which each decision tree makes a prediction (votes) for the

class of a given input data point. The **final class prediction** is determined by a majority vote:

$$\text{Final Class} = \text{mode of all tree predictions} \tag{5}$$

The class that receives the most votes from the individual trees is chosen as the final output.

E. Boosting

The decision tree represents the fundamental learner in the boosting process, which can be enhanced by addressing its inherent limitations through appropriate corrective measures.

- Gradient Boosting using $entropy(S)$ as the loss function, to minimize it, the algorithm begins by initializing the model with a simple prediction, typically the **logits** (untransformed outputs)

$$F_k^{(0)}(x_i) = 0 \quad \text{for each class } k \tag{6}$$

for each class. For multi-class classification, a common loss function used is the Softmax Cross-Entropy Loss

$$L(y_i, \mathbf{F}(x_i)) = -\sum_{k=1}^{K} y_{ik} \log\left(p_k(x_i)\right) \tag{7}$$

,where, y_i is true class label for the i-th data point, represented as a one-hot encoded vector, $p_k(x_i)$ is the predicted probability for class k, computed using the softmax function:

$$p_k(x_i) = \frac{\exp(F_k(x_i))}{\sum_{j=1}^{K} \exp(F_j(x_i))} \tag{8}$$

and $F_k(x_i)$ is the logit (raw score) output for class k at data point x_i, and K is the total number of classes. This gradient g_{ik} indicates how much the prediction for class k needs to be adjusted to reduce the loss:

$$g_{ik} = \frac{\partial L(y_i, \mathbf{F}(x_i))}{\partial F_k(x_i)} = p_k(x_i) - y_{ik} \tag{9}$$

and y_{ik} is the true label indicator (1 if $y_i = k$, 0 otherwise). In the end, the model is updated by adding the output of the new trees to the existing logits for each class:

$$F_k^{(m)}(x_i) = F_k^{(m-1)}(x_i) + \eta h_k^{(m)}(x_i) \tag{10}$$

, where η is the Learning rate, which controls the size of the update step and $h_k^{(m)}(x_i)$ is the prediction from the m-th tree for class k. Furthermore, among those processes, a separate decision tree is trained for each class k using the computed gradients g_{ik} as the target values. The goal is for each tree to predict the adjustments needed to minimize the loss and the output of each tree $h_k^{(m)}(x_i)$ represents the adjustments for the logits for class k.

- XGBoost (Extreme Gradient Boosting) is founded upon the concept of boosting, which entails the consolidation of a set of relatively weak models (typically decision trees) into a unified and robust predictor. This is achieved by sequentially training the models, with each subsequent model focused on correcting the errors identified in the previous models. The introduction of L1 (Lasso) and L2 (Ridge) regularization terms into the loss function to consist Objective function

$$\text{Objective} = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{T} \Omega(f_k) \tag{11}$$

serves to prevent overfitting, thereby enhancing the robustness of the method in question when compared to traditional gradient-boosting techniques. Additionally, the **second-order Taylor approximation** of the objective function is used to simplify the optimization:

$$\text{Obj} \approx \sum_{i=1}^{n} \left[g_i f(x_i) + \frac{1}{2} h_i f(x_i)^2 \right] + \Omega(f) \tag{12}$$

where g_i is the First-order gradient of the loss function and h_i: Second-order gradient (Hessian).
Then, To decide where to split the tree, XGBoost calculates the **gain** from potential splits. The gain measures how much a split improves the model:

$$\text{Gain} = \frac{1}{2} \left[\frac{\left(\sum_{i \in I_L} g_i\right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i\right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i\right)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma \tag{13}$$

where I_L and I_R are the sets of data points in the left and right child nodes and λ and γ are regularization parameters that penalize tree complexity to prevent overfitting. For each leaf of the tree, XGBoost calculates the **optimal weight** using:

$$w_j = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{14}$$

The process is repeated with each new tree, with the residual errors of the previous iteration serving as the focus. This continues until the model achieves the desired level of performance or reaches the maximum number of iterations.

F. Neural Network

- Calculate the error at the output layer. $\delta^{(l)}$ is the error term of the output layer, $f'\left(z^{(l)}\right)$ is the derivative of the activation function at the l layer.

$$\delta^{(L)} = \frac{\partial L}{\partial a^{(L)}} \cdot f'\left(z^{(l)}\right) \tag{15}$$

- The error is then backpropagated to the previous layer, calculating the error for each layer in turn. W is the weight vector, $\left(W^{(l+1)}\right)^T \delta^{(l+1)}$ is the backpropagated error from the $(l+1)$ layer.

$$\delta^{(l)} = \left(W^{(l+1)}\right)^T \delta^{(l+1)} \cdot f'\left(z^{(l)}\right) \tag{16}$$

- Calculate weights and bias gradients. $a^{(l-1)}$ is the transpose of the activation from the $(l-1)$ layer

$$\frac{\partial L}{\partial W^{(l)}} = \delta^{(l)} \cdot \left(a^{(l-1)}\right)^T \tag{17}$$

$$\frac{\partial L}{\partial b^{(l)}} = \delta^{(l)} \tag{18}$$

- Using Stochastic Gradient Descent to update weights and bias:

$$W^{(l)} = W^{(l)} - \eta \frac{\partial L}{\partial W^{(l)}} \tag{19}$$

$$b^{(l)} = b^{(l)} - \eta \frac{\partial L}{\partial b^{(l)}} \tag{20}$$

G. Optimizers

- Stochastic Gradient Descent (SGD) is a method that randomly extracts a sample from the training set to calculate the gradient of the loss function, which can greatly improve the running speed:

$$\theta_t = \theta_{t-1} - \eta \cdot \nabla_\theta L(\theta; x^{(i)}, y^{(i)}) \tag{21}$$

  where t is the iteration number, η is the learning rate, and $\nabla_\theta L(\theta; x^{(i)}, y^{(i)})$ is the gradient of the loss function with respect to the parameter θ.
- Adam (Adaptive Moment Estimation) combines the advantages of momentum and adaptive learning rates. To process it, we use

$$\theta_{t+1} = \theta_t - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{22}$$

  to update parameters. α: Learning rate and ϵ: Small constant to prevent division by zero. And the m_t and v_t are updated by **First Moment (Mean):**

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{23}$$

  and **Second Moment (Variance):**

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{24}$$

  where g_t is the gradient of the loss function and the β_1 and β_2 are the decay rate that controls how much past gradients influence the current variance estimate.

H. L2 Regularization and Dropout

- L2 Regularization is used to modify loss function with

$$\text{Loss} = \text{Original Loss} + \lambda \sum_i w_i^2 \tag{25}$$

  where λ is a hyperparameter that controls how much to penalize large weights and w_i: Weights of the model.It encourages the model to keep the weights smaller, leading to a simpler and more generalizable model.
- Dropout is used in the neural network, in each training iteration, a random subset of neurons is disregarded (or "dropped out"), and only the remaining neurons are utilized (We can set the using rate to find the best one)

I. workflow Diagram

Figure 10 shows our overall process of testing our model on datasets.

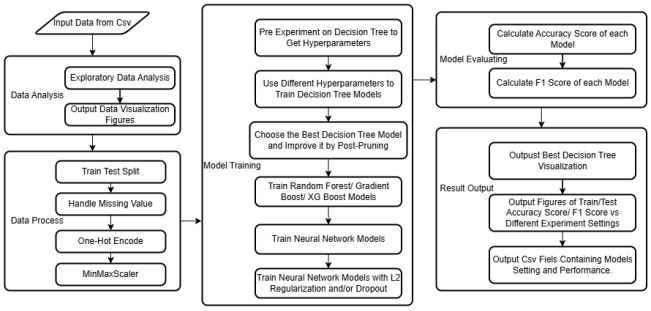


Fig. 10: Workflow

J. Software Suite

Libraries and Frameworks: We use `Scikit-Learn` for traditional ML models and `TensorFlow` for Neural Networks. We use `xgboost` for XG Boost. We use `numpy` and `pandas` for data process. We use `json` and `sys` to manage the files. **Metrics**: We evaluate model performance using `accuracy_score`, `f1_score`, `classification_report`, `confusion_matrix`, and visualization tools like `ConfusionMatrixDisplay`.

K. Experiment Setting

- **Training and Testing Split**: We use an 60/40 split for training and testing sets. The dataset is preprocessed to handle missing values and categorical data. Missing values are handled using `SimpleImputer` with median imputation for numerical features and the most frequent imputation for categorical features.
- **Multiple Experimental Runs**: We split each dataset 10 times and train various types of models on each of these training sets. When calculating the metrics like accuracy and F1 score, we take the average over the ten experiments.
- **Decision Tree**
  - Hyperparameter Search:
    We use `RandomizedSearchCV` to optimize the Decision Tree parameters over 400 iterations with 8-fold cross-validation. The hyperparameters explored include:
    * `max_depth`: 10, 20, 30, 40, no restriction
    * `min_samples_split`: 2, 5, 10, 20
    * `min_samples_leaf`: 1, 2, 5, 10
    * `criterion`: gini, entropy
  - Post-Pruning:
    We perform post-pruning using cost-complexity pruning to determine the optimal `ccp_alpha` value.
- **Random Forest**
  - We use the `RandomForestClassifier` within up to 200 estimators, recording the changes of accuracy along with the increasing number of estimators.
  - Number of Estimators: 2, 3, 5, 7, 11, 13, 20, 30, 50, 70, 90, 120, 150, 200.

- **Gradient Boosting**
  - Model: `GradientBoostingRegressor`
  - Hyperparameters:
    - `Learning Rate`: 0.02, 0.05, 0.1, 0.3
    - `Number of Estimators`: 2, 3, 5, 7, 11, 13, 20, 30, 50, 70, 90, 120, 150, 200.
- **XGBoost Classifier**
  - Model: `XGBClassifier`
  - Hyperparameters:
    - `n_estimators`: 2, 3, 5, 7, 11, 13, 20, 30, 50, 70, 90, 120, 150, 200.
    - `reg_lambda`: 0.02, 0.1, 1, 3, 5, 9, 15.
    - `learning_rate`: Fixed at 0.3.
- **Neural Network**
  - Layers:
    - Input Layer: Matches the number of input features.
    - Hidden Layers: Two hidden layers, each with $\sqrt{D_i \times D_o}$ neurons, where D_i is the input dimension and D_o is the output dimension.
    - Output Layers: Matches the number of output classes.
    - Activation: 'ReLU' activation for hidden layers and 'Softmax' for the output layer.
  - L2 Regularization with a penalty of: 1e-5, 3.16e-5, 1e-4, 3.16e-4, 1e-3.
  - Dropout layer is added with a dropout rate specified by `dropout_parameter`: 0.2, 0.4, 0.6, 0.8.
  - Optimizer: Adam and SGD optimizer is used by default, with `categorical_crossentropy` loss.
  - Early Stopping: Monitors validation loss, with patience set to 5 epochs and restoration of the best weights.

III. Result

After applying our methodology to three different datasets, we get the following results.

A. Decision Tree

- Among all the groups of hyperparameters that have been examined, we report 10 of them, as they perform better relatively. The maximum depth parameter is of great consequence, as a deeper tree can learn more intricate patterns; however, this also renders it more susceptible to overfitting, whereby the model erroneously identifies noise as meaningful relationships. Table I demonstrates that models with greater depth often exhibit higher training accuracy (up to 0.8141), yet occasionally show difficulty in generalizing, as evidenced by lower test accuracies. The training accuracies ranged from 0.7458 to 0.8141, indicating that some models were highly tuned to the training data. In contrast, the test accuracies were between 0.5622 and 0.5816, demonstrating the models' capacity for generalization to unseen data. The highest test accuracy, 0.5816, is achieved using the "Gini" criterion with a depth of 40, a minimum of 10 samples per leaf, and a minimum split requirement of 2 samples.

TABLE I: Decision Tree Accuracy Metrics

Criterion	Depth	Leaf	Split	Acc Train	Acc Test
entropy	40	5	5	0.8141	0.5622
entropy	35	2	8	0.8123	0.5635
entropy	25	7	12	0.8055	0.5671
entropy	20	3	15	0.8092	0.5698
Gini	50	8	3	0.7901	0.5754
Gini	45	4	10	0.7998	0.5782
Gini	40	10	2	0.7458	0.5816
Gini	30	5	10	0.8133	0.5628
Gini	30	1	20	0.7968	0.5689
Gini	10	1	5	0.8105	0.5801

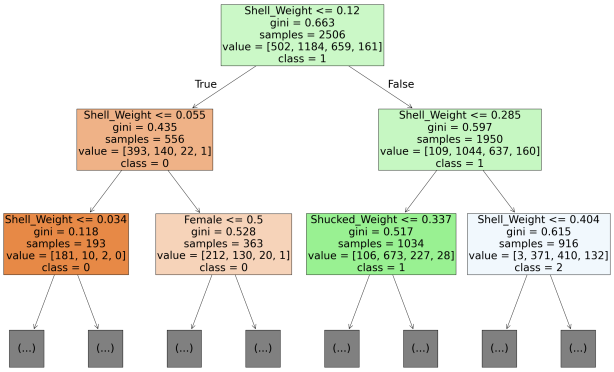


Fig. 11: First Three Rows of The Tree

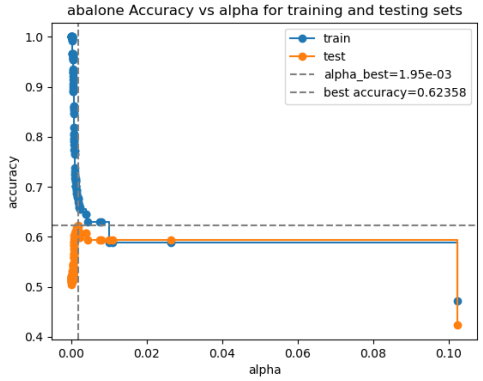


Fig. 12: Performances of Post-pruning with different α on Abalone set

Furthermore, we visualized the total Decision tree in the appendix, as it is too large to show in the paper. It shows all the nodes and leaves of the tree. For example, the decision tree commences with "Shell Weight" at the root node. If the "Gini" value of "Shell Weight" is less than or equal to 0.12, the decision tree diverges to the left, if smaller than 0.02 continues to the right. In the left branch, the tree proceeds to examine "Shell Weight" at a threshold of 0.055 (Gini). If this value is less than or

equal to 0.055, the path continues to the left; otherwise, it shifts to the right. On the right side of the root node, the tree then checks if "Shell Weight" is less than or equal to 0.285. If so, it leads to the left; if not, it branches to the right, shown in Figure 11.

- Post-pruning

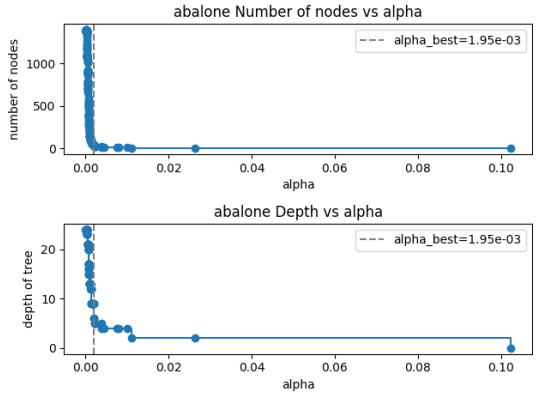


Fig. 13: Complexities of Post-pruning with different α on Abalone set

Figure 12 illustrates the performance of using Post-pruning to find the best hyperparameters, as the increase of α. We found the best number of α is 1.95e-03 which trade-off between the tree's complexity and its accuracy and we got the best performance on accuracy(0.62585). Hence, post-pruning outperforms our setting groups of hyperparameters increasing by almost 0.04 in accuracy. Furthermore, 13 reveals that a relatively modest alpha value maintains the tree's complexity, whereas a larger alpha value facilitates a more pronounced pruning of the tree, simplifying the model and solving overfitting.

Based on the above analysis, Post-pruning performance is better when finding the best hyperparameter, which can improve the model further compared to the manual settings. Furthermore, we got an accuracy 0.62585 when using Post-pruning.

B. Random Forest

Figure 14 demonstrates the influence of the number of estimators (trees) on the precision of a Random Forest model for both training and testing data. The training accuracy, represented by the blue curve, exhibits a rapid increase and then reaches a plateau close to 1.0 as the number of trees increases. This rapid ascent demonstrates that the Random Forest model effectively captures intricate patterns within the training data, thereby reducing bias and ensuring an optimal fit. In contrast, the testing accuracy, illustrated by the orange curve, commences at a considerably lower level and subsequently increases gradually with the addition of more trees. The curve subsequently reaches a plateau at approximately 0.62412, indicating enhanced generalization with an increasing number of trees. However, this also demonstrates a point

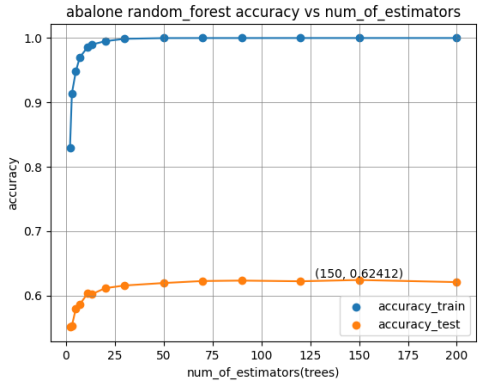


Fig. 14: Random Forest with different number of Trees on Abalone set

of diminishing returns, with minimal improvement in test accuracy observed in 150 estimators. Meanwhile, all perform well when trade-off precision and recall, as it gets a good F1 score (0.61544), shown in Figure 15

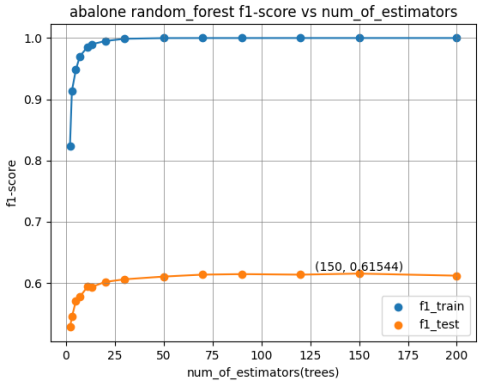
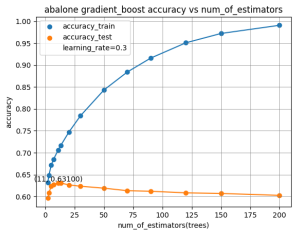


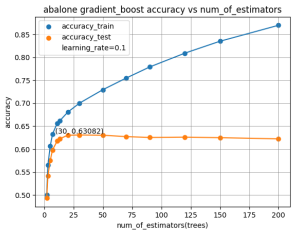
Fig. 15: F1 Score of Random Forest with different number of Trees on Abalone set

C. Boosting

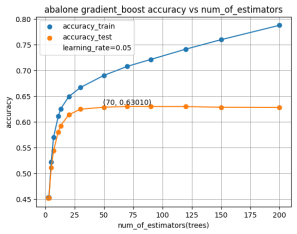
- Gradient Boosting
  Figure 16 demonstrates the impact of varying the learning rate on the performance of a gradient boosting model as the number of estimators (trees) increases. The training accuracy demonstrates a consistent improvement as the number of estimators is increased across all graphs, except at a learning rate of 0.3, the test accuracy rapidly reaches a peak of approximately 0.5, indicating poor generalization due to overfitting(Figure16(a)). In contrast, lower learning rates, such as 0.02, result in slower, more gradual increases, reflecting a more conservative approach to learning(Figure 16(d)), which gets the highest accuracy among those learning rates (0.63106) and learning rates 0.1 and 0.02 perform similarly, with around 0.63 slightly lower than 0.02. Furthermore, f1 scores also perform well and reach their highest value (Figure 17)



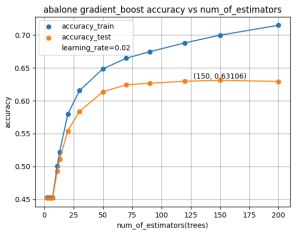
((a)) learning rate = 0.3



((b)) learning rate = 0.1

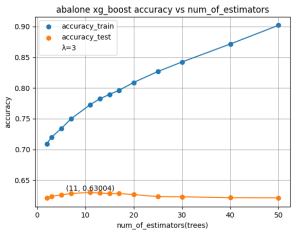


((c)) learning rate = 0.05

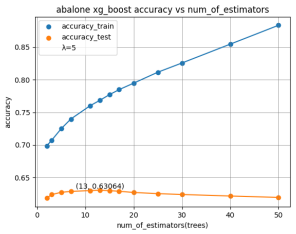


((d)) learning rate = 0.02

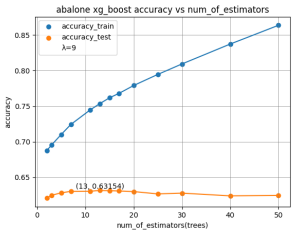
Fig. 16: Testing of the different learning rates of Gradient Boosting on Abalone set



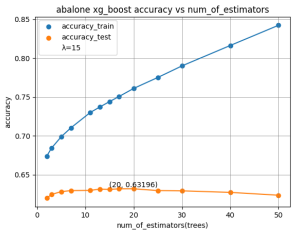
((a)) Decay rate = 3



((b)) Decay rate = 5



((c)) Decay rate = 9



((d)) Decay rate = 15

Fig. 18: Testing of the different Decay rates in XGBoost on Abalone set

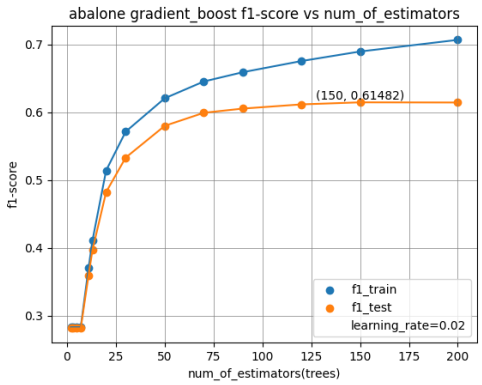


Fig. 17: F1 Score when Learning rate = 0.02 on Abalone set

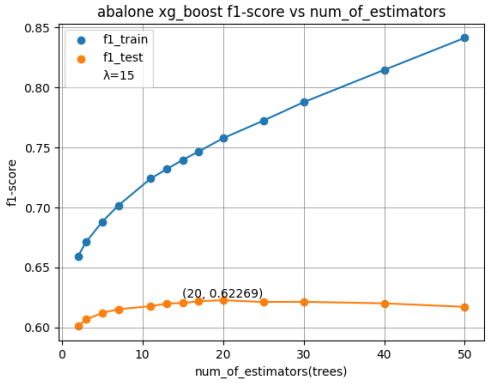


Fig. 19: F1 Score when Deacy Rate = 15 on Abalone set

- XGBoost
  Figure 18 shows the influence of varying decay rates (λ) on the efficacy of training and test accuracy. As the number of trees increases, the blue curve representing training accuracy continues to rise, while the orange curve representing test accuracy stabilizes at varying levels contingent on the decay rate. In the case of a high decay rate of 15, the training accuracy demonstrates a gradual increase, stabilizing at approximately 0.72. In contrast, the test accuracy ultimately reaches a value around 0.632, when 20 estimators. A reduction in the decay rate results in an increase in the discrepancy between the model's performance on the training set and the test set. Further analysis demonstrated that the optimal decay rate was 15 across all the examined cases.

Following a comparative analysis of Gradient Boosting and XGBoost, it was evident that XGBoost exhibited superior performance and accuracy compared to Gradient Boosting, and for XGBoost setting the Decay rate as 15 can be the best choice. Moreover, we got an accuracy of 0.632 when using XGBoost.

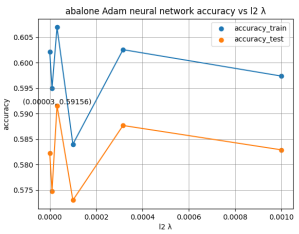
D. Simple Neural Network

In this section, we compare the performance of Adam and SGD(optimizers) under the same setting of NN.

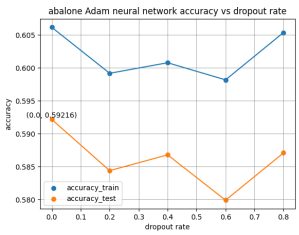
TABLE II: Best Performances of Models on Abalone Set

Model Name	Acc Train	Acc Test	F1 Train	F1 Test
Adam	0.6067	0.5987	0.5141	0.4923
SGD	0.6069	0.5877	0.4630	0.4393

- **Comparison of Adam and SGD** After the experiments, we observe, from Table II, that Abam (0.5987) performs



((a)) Performance of NN with different Decay Weight



((b)) Performance of NN with different Dropout Rate

Fig. 20: Comparison between Dropout and L2 Regularization with accuracy on Abalone set

slightly better than the SGD in accuracy on the test set, as well as the performance of F1 Score, Adam performs much better when trade-off precision and recall, however, both of them doing worse. So, NN may not be suitable to utilize in this classification problem.

- **Comparison of Dropout and L2 Regularization** Table III shows the accuracy of using 4 different groups of Decay Weight and Dropout rate, for the first two and the last group, we find F1 score on the test set is lower than 0.5, which illustrates it cannot find the balance between Precision and recall. Meanwhile, we got an accuracy on the test set of 0.5898 when using the Decay Weight of 0.0001 and Dropout rate of 0.7, f1 also larger than 0.5 on the test set. Furthermore, after we test the different Dropout rates and L2 regularization separately, we find they just perform similarly on abalone, and the highest accuracy (0.59156) is reached by L2 regularization(Decay Weight = 0.00003) among the examined hyperparameters, shown in Figure 20

TABLE III: Neural Network Performance Metrics Based on Adam with L2 regularization and Dropout on Abalone set

L2	Dropout	Acc Train	Acc Test	F1 Train	F1 Test
0.0001	0.1	0.5986	0.5841	0.4702	0.4466
0.0001	0.4	0.5882	0.5580	0.4831	0.4444
0.0001	0.7	0.6057	0.5898	0.5234	0.5002
0.001	0.1	0.6043	0.5847	0.5026	0.4806

TABLE IV: Model Performance Metrics on Abalone set

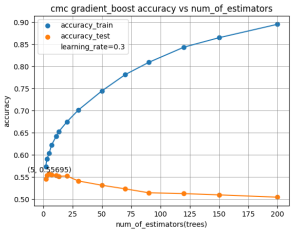
Model Name	Acc Train	Acc Test	F1 Train	F1 Test
Adam	0.6067	0.5987	0.5141	0.4923
SGD	0.6069	0.5877	0.4630	0.4393
Adam(Dropout and L2)	0.6057	0.5898	0.5234	0.5002
Decision Tree	0.6717	0.6259	0.6616	0.6125
Random Forest	1.0000	0.6241	1.0000	0.6154
Gradient Boost	0.6998	0.6311	0.6899	0.6148
XGBoost	0.7611	0.6320	0.7577	0.6227

After finishing all the experiments of models with different hyperparameters, two of the best models were decided. Table IV reveals the best performance of different models under our methodology, Gradient Boost, and XGBoost ranking as the first two models with accuracy both over 0.631. So in the next section, we apply those two models again on cmc set to test if it still doing well.

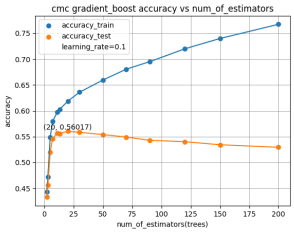
E. Apply Best two Models on CMC Set

In this section, we apply the best two models on the Abalone set to the CMC set, to test their performance.

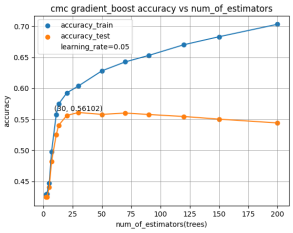
- Gradient Boosting Under the different learning rates, Figure 21 demonstrates that Gradient Boosting always shows overfitting as the increase of the number of estimators. Also, the accuracy decreases continuously after reaching the maximum value and the gap in accuracy between the test set and train set keeps increasing, except for the learning rate of 0.02. However, The best performance (0.56102) among those learning rates of Gradient Boosting is obtained with a learning rate of 0.05, observed in Figure 21(a)



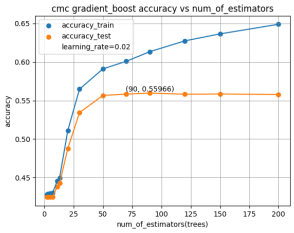
((a)) Learning rate = 0.3



((b)) learning rate = 0.1



((c)) learning rate = 0.05



((d)) learning rate = 0.02

Fig. 21: Testing of the different Decay rates in Gradient Boosting on CMC set

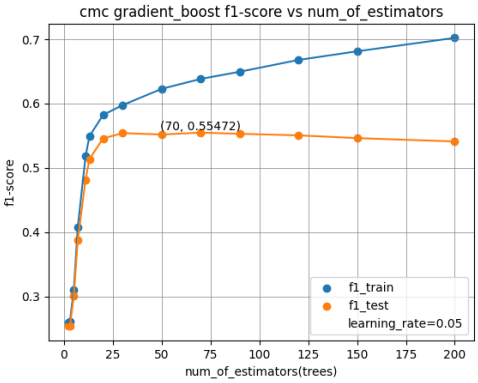
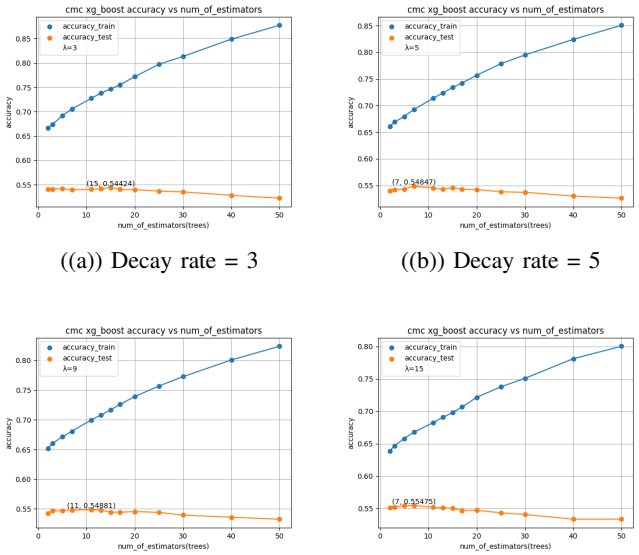


Fig. 22: F1 Score when learning Rate = 0.05 on CMC set with Gradient Boosting

Moreover, finding in Figure 22the performance of F1 also reaches our level trade-off of the value of Precision and Recall when the learning rate of 0.05.

- XGBoost
  We observe the output of XGBoost with four different Decay rates in Figure 23 that illustrates the increase in Decay rate accompanied by the increase in accuracy. The Decay rate of 15 gives us the best accuracy (0.55475) within only 7 trees, converging faster than on the abalone set (Figure 23(d)). The F1 score of Decay rate of 15 also trade-off the precision and recall well (24) Moreover, same with Gradient Boosting, XGBoost also shows a trend of overfitting with the increase of estimators.



((a)) Decay rate = 3 ((b)) Decay rate = 5

((c)) Decay rate = 9 ((d)) Decay rate = 15

Fig. 23: Testing of the different Decay rates in XGBoost on CMC Set

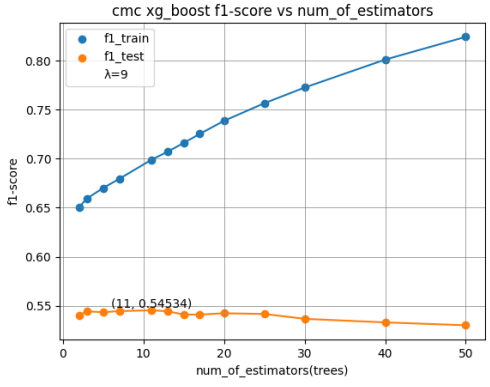


Fig. 24: F1 Score when Deacy Rate = 15 on CMC with XGBoost

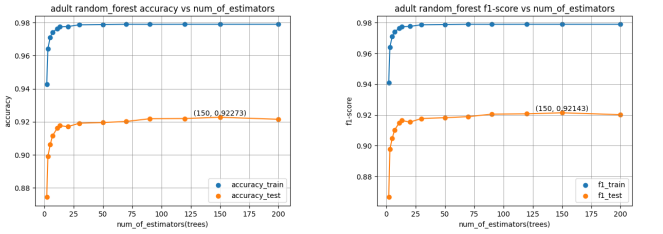
After applying the best two models found in the abalone set to the cmc set, we find Gradient Boosting performs slightly better than XGBoost with an accuracy of 0.56102, which also reveals that on this dataset boosting can not obtain a good classification as the super low accuracy for a triple classification. Meanwhile, both of them have a high rest of overfitting depend on the number of estimators.

F. Find the Best model for Adult Set with Same Methodology

In this section, we repeat the same methodology on the Adult set (Large dataset), to examine the best model on it.

TABLE V: Best Performances of Models on Adult Set

Model Name	Acc Train	Acc Test	F1 Train	F1 Test
Adam	0.8802	0.8704	0.8780	0.8678
SGD	0.8773	0.8706	0.8780	0.8678
Decision Tree(Gini)	0.9181	0.8724	0.9158	0.8689
Gradient Boost	0.8937	0.8766	0.8910	0.8733
XGboost	0.8895	0.8724	0.8861	0.8681
Adam(L2 and Dropout)	0.8812	0.8742	0.8774	0.8693
Random Forest	0.9790	0.9227	0.9790	0.9214



((a)) Random Forest ((b)) F1 score of Random Forest

Fig. 25: Performance of Random Forest on Adult set

Table V shows the performance of different models with the best hyperparameters of them under our methodology. Among the Random Forest gained the highest accuracy (0.9227) and F1 score (0.9214), showing that is the best model on Adult Set compared to other models we used and in Figure 25, the random forest gets the best accuracy with the same number of estimators as on the abalone set. Except for Random Forest, other models have similar performance, with an accuracy of around 0.87 and an F1 score of around 0.86. Among these, Neural Network got the lowest accuracy (0.8704) and F1 score (0.8678).

IV. Discussion

A. Summary

The objective was successfully attained, with satisfactory responses obtained in the classification of specimens. The specimens were classified into the following categories: Class 1 (0–7 years), Class 2 (8–10 years), Class 3 (11–15 years), and Class 4 (Larger than 15 years). Furthermore, an overall comparison has been conducted between each model on the various datasets (of varying sizes) previously mentioned. The application of boosting represents the optimal methodology for the classification of the abalone set, encompassing four distinct classes. The performance of

XGBoost exhibits a slight advantage over Gradient Boosting, with an accuracy of approximately 0.632 on the test set. Both approaches demonstrate proficiency in achieving a balance between precision and recall, as evidenced by the F1 score. Moreover, the two models were tested on a very small dataset (CMC set), with gradient boosting demonstrating superior accuracy (0.56102) on the test set. The methodology previously outlined was then applied to a large dataset (Adult set). The results demonstrated that Random Forest exhibited the most optimal performance, achieving both high accuracy (0.9227) and an F1 score (0.9214) on the test set. This suggests that Random Forest is more effective in large datasets while Boosting is more suitable for relatively small datasets.

B. Limitation

After completing all the experiments and summarizing the output, we found some main limitations with the experiments. For the abalone dataset as an imbalanced classification dataset, the presence of imbalanced training data has a significant detrimental effect on the performance of any given system [15]. The predominant machine learning technique addresses the issue of imbalanced datasets by prioritizing the avoidance of the minority class and the reduction of inaccuracy in the majority class [15]. One of the widely used methods is the SMOTE algorithm, which generates new samples by interpolating the neighboring samples of the minority class samples. However, it may cause the generated data to deviate from the minority class center. Moreover, the SMOTE method improved by the normal distribution will control the location of the new sample generation to be closer to the minority class center and optimize the classification effect [16]. We can test both of them, finding the best before applying our models.

When considering the CMC dataset, which only contains roughly 1400 samples, it is undoubtedly a small dataset. Neural networks, known for their data-greedy nature, observe their predictive ability is significantly affected when trained on limited data, using them with small datasets requires caution to ensure accurate and reliable predictions [17]. One promising solution is the use of pre-trained networks, a technique known as transfer learning. This method initializes the neural network with weights pre-trained in a related domain and then fine-tunes the model using the specific data of interest [18]. By starting from a well-initialized state, this approach can facilitate more effective training, especially when the domains share similarities [18]. If relevant data or the original, larger CMC dataset were available, training on it and applying transfer learning could significantly enhance the neural network's predictive performance [19] [20].

Furthermore, due to the limitation of hardware performance and time limitations, we may not well train the neural network, which will affect its predictive ability. If other variants of Adam optimizer, such as AdaBound, NosAdam, and Yogi [21] or some other types like Nesterov Accelerated Gradient (NAG), are adopted, the convergence of loss function can be accelerated [22] and stabilized, hence the neural network may achieve better performance. Furthermore, when applying the neural network to the adult set, an upper limit of 90 iterations was set, and improvements of less than 0.001 were regarded as insignificant in the early stopping algorithm, which would result in varying degrees of underfitting on large datasets. This insufficient running significantly impacts the neural network model's ability to predict this complex problem [23], which can seen as one of the reasons for the relatively worse performance of Neural Networks on the adult set.

In the end, when we determined that random forests are more appropriate for large datasets and that boosting is more suitable for small datasets, we did not conduct comprehensive experiments on a multitude of datasets. Our experiments may not be entirely precise, and additional experiments are still necessary to corroborate this assertion.

V. Conclusion

In our research, we determined the best two models for the classification of abalone age, which help fishers easier to make a price for the abalone. We also observed that for classification problems, to fit small datasets may prefer to use the boosting method, however, when dealing with large data problems, the random forest can be the better one, among the examined models, including Decision Tree(with post-pruning or not), Random Forest, Gradient Boosting, XGBoost, and simple neural network. Nevertheless, there are still a lot of limitations that still need to be fixed. For future research, we can improve our model further, as mentioned in the limitation part. There are still many other good models for small/large datasets. we can test their performance to gain a better choice when confronting different types of problems and also, try to find some general methods, that have a super ability on all sizes of datasets. For instance, a deep convolutional neural network (CNN) can be effectively employed to accommodate modest datasets, provided that appropriate modifications are implemented [24]. It is not necessary to resort to the redesign of bespoke, smaller networks, contending that a model that is sufficiently robust to accommodate a large dataset can also be applied to a smaller one [24].

References

[1] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *Journal of Chemometrics*, vol. 18, no. 6, pp. 275–285, 2004. doi: 10.1002/cem.873.

[2] Y. Xia, X. Guo, Y. Li, L. He, and X. Chen, "Deep learning meets decision trees: An application of a heterogeneous deep forest approach in credit scoring for online consumer lending," *Journal of Forecasting*, vol. 41, no. 8, pp. 1669–1690, 2022. doi: 10.1002/for.2891.

[3] M. M. Jovanovic, L. Kascelan, M. Joksimovic, and V. Kascelan, "Decision tree analysis of wine consumers' preferences: evidence from an emerging market," *British Food Journal*, vol. 119, no. 6, pp. 1349–1361, 2017. doi: 10.1108/BFJ-11-2016-0568.

[4] S. A. Naghibi, K. Ahmadi, and A. Daneshi, "Application of Support Vector Machine, Random Forest, and Genetic Algorithm Optimized Random Forest Models in Groundwater Potential Mapping," *Water Resources Management*, vol. 31, no. 9, pp. 2761–2775, 2017. doi: 10.1007/s11269-017-1660-3.

[5] L. Cheng, J. De Vos, P. Zhao, M. Yang, and F. Witlox, "Examining non-linear built environment effects on elderly's walking: A random forest approach," *Transportation Research Part D: Transport and Environment*, vol. 88, Art. no. 102552, 2020. doi: 10.1016/j.trd.2020.102552.

[6] K. Esfandiari, F. Abdollahi, and H. A. Talebi, *Neural Network-Based Adaptive Control of Uncertain Nonlinear Systems*, 1st ed., Springer International Publishing, 2022. doi: 10.1007/978-3-030-73136-6.

[7] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms*, 1st ed. The MIT Press, 2012. doi: 10.7551/mitpress/8291.001.0001.

[8] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, vol. 1168, no. 2, Art. no. 022022, Feb. 2019. doi: 10.1088/1742-6596/1168/2/022022.

[9] R. N. D'souza, P.-Y. Huang, and F.-C. Yeh, "Structural analysis and optimization of convolutional neural networks with a small sample size," *Scientific Reports*, vol. 10, no. 1, Art. no. 834, 2020. doi: 10.1038/s41598-020-57866-2.

[10] B. P. Ooi *et al.*, "Random subspace oracle (RSO) ensemble to solve small sample-sized classification problems," *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 4, pp. 3225–3234, 2019. doi: 10.3233/JIFS-18504.

[11] R. Andonie, "Hyperparameter optimization in learning systems," *Journal of Membrane Computing*, vol. 1, no. 4, pp. 279–291, Dec. 2019. doi: 10.1007/s41965-019-00023-0.

[12] Chengyuan Zhang, "Using Abalone's Physical Features to Predict its Age," *Highlights in Science, Engineering and Technology*, vol. 47, pp. 95–105, May 2023. doi: 10.54097/hset.v47i.8171.

[13] Seda Guney, Irem Kilinc, Alaa Ali Hameed, and Akhtar Jamil, "Abalone Age Prediction Using Machine Learning," in *Pattern Recognition and Artificial Intelligence*, C. Djeddi, I. Siddiqi, A. Jamil, A. A. Hameed, and I. Kucuk, Eds., Springer International Publishing, Cham, 2022, pp. 329–338. doi: 10.1007/978-3-031-04112-9_25.

[14] Y. Song and Y. Lu, "Decision tree methods: applications for classification and prediction," *Shanghai Archives of Psychiatry*, vol. 27, no. 2, pp. 130–135, Apr. 2015. doi: 10.11919/j.issn.1002-0829.215044.

[15] P. Kumar, R. Bhatnagar, K. Gaur, and A. Bhatnagar, "Classification of imbalanced data: Review of methods and applications," *IOP Conference Series: Materials Science and Engineering*, vol. 1099, no. 1, Art. no. 012077, Mar. 2021. doi: 10.1088/1757-899X/1099/1/012077.

[16] S. Wang, Y. Dai, J. Shen, and J. Xuan, "Research on expansion and classification of imbalanced data based on SMOTE algorithm," *Scientific Reports*, vol. 11, no. 1, p. 24039, 2021. doi: 10.1038/s41598-021-03328-2.

[17] R. Lanouette, J. Thibault, and J. L. Valade, "Process modeling with neural networks using small experimental datasets," *Computers & Chemical Engineering*, vol. 23, no. 9, pp. 1167–1176, 1999. doi: 10.1016/S0098-1354(99)00282-3.

[18] Rhett N. D'souza, Po-Yao Huang, and Fang-Cheng Yeh, "Structural Analysis and Optimization of Convolutional Neural Networks with a Small Sample Size," *Scientific Reports*, vol. 10, no. 1, p. 834, 2020. doi: 10.1038/s41598-020-57866-2.

[19] Y. Sawada and K. Kozuka, "Whole layers transfer learning of deep neural networks for a small scale dataset," *International Journal of Machine Learning and Computing*, vol. 6, no. 1, p. 27, 2016.

[20] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020. doi: 10.1109/JPROC.2020.2991924.

[21] D. Soydaner, "A comparison of optimization algorithms for deep learning," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 34, no. 13, p. 2052013, 2020. doi: 10.1142/S0218001420520133.

[22] W. Cheng, X. Yang, B. Wang, and W. Wang, "Unbiased quasi-hyperbolic Nesterov-gradient momentum-based optimizers for accelerating convergence," *World Wide Web*, vol. 26, no. 4, pp. 1323–1344, Jul. 2023. doi: 10.1007/s11280-022-01086-3.

[23] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *arXiv preprint arXiv:1804.07612*, 2018.

[24] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *Proc. 3rd IAPR Asian Conf. Pattern Recognition (ACPR)*, Kuala Lumpur, Malaysia, 2015, pp. 730–734. doi: 10.1109/ACPR.2015.7486599.

APPENDIX A
VISUALIZATION OF DECISION TREE

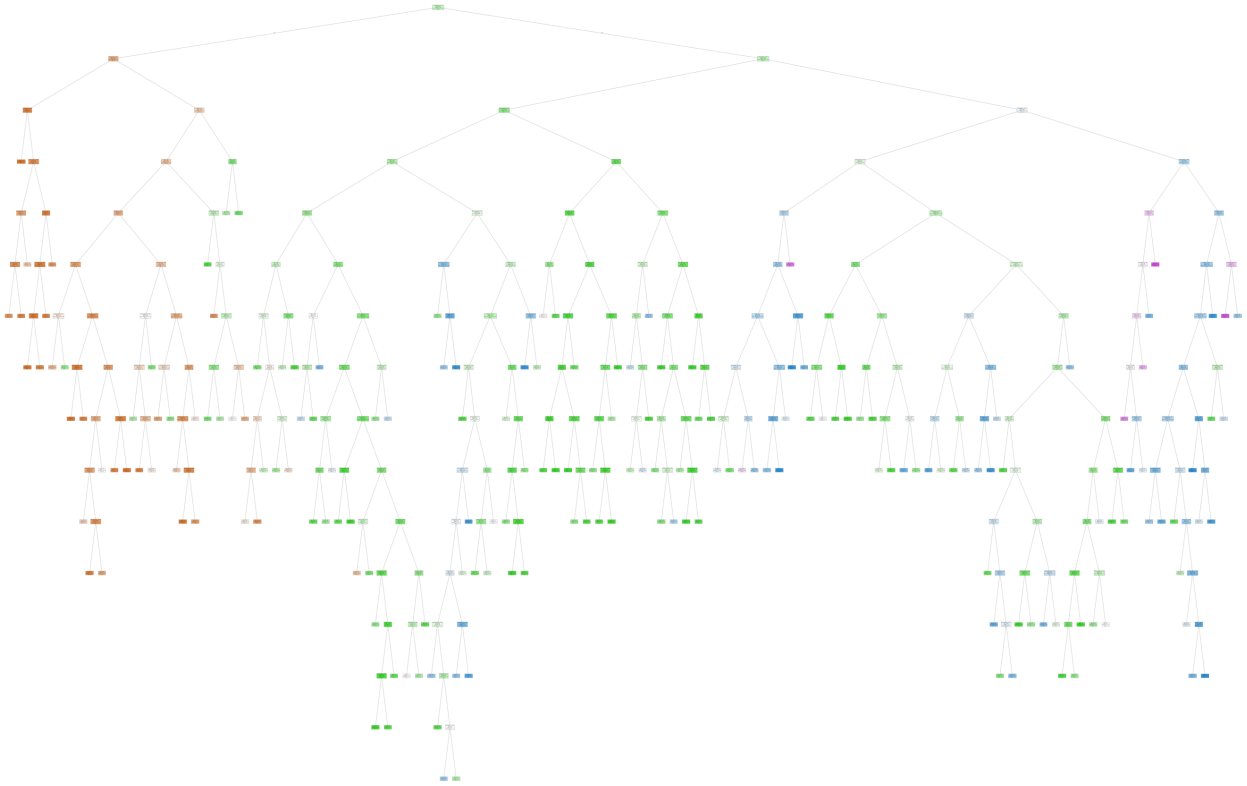


Fig. 26: Visualization of the Tree