Application of Decision Trees and Random Forests to Predict the Age of Abalone

Author: Tarunroop Dhillon

***Abstract*—Abalone age prediction based on its physical characteristics is a widely studied problem which has seen numerous different learning techniques being applied. One reason for this interest is the commercial value of Abalone and the role age plays in determining this value. In this paper we investigate the application of classification Decision Trees (more specifically the CART algorithm implementation) and Random Forests to determine the age of Abalone. The performance of decision trees is analyzed both with and without regularization. Two different techniques of regularizing are investigated, the first being an early stopping technique using hyperparameters to control the tree size during the fitting process and the second technique being a post-pruning technique. Random Forests are subsequently used to investigate if they increase the performance as compared to Decision Trees. Exploratory data analysis and visualization are used in the first instance to develop key insights into the Abalone dataset. We found that regularizing the Decision Trees improved the classification performance from 53% to 63% on the test set, however Random Forests were not able to provide further improvements on the test set. The feature importance scores resulting from fitting these models are also discussed.**

Keywords—abalone, decision trees, random forests, cost-complexity pruning, feature importance

I. Introduction

Abalone are a variety of shellfish that are widely harvested through aquaculture farms in many countries. The meat of Abalone is highly nutritious and is considered a delicacy in many parts of the world. Further, Abalone shells have commercial value as decor and are also farmed as a source of mother-of-pearl [1]. Due to the commercial value of Abalone and to support sustainable farming practices, determining the age of Abalone is very important. The established procedure of determining the age involves cutting the Abalone's shell, staining it, and counting the number of rings through a microscope. The number of rings plus 1.5 is generally considered to be the rough age of Abalone. This process is however slow and tedious. An alternative is to use the physical measurements of Abalone such as height, diameter, length etc. to determine the age. A famous dataset that provides these physical measurements and the one we use in this paper is the widely known 'Abalone dataset' available on the UCI Machine Learning Repository [2].

Decision trees have come to be one of the most popular machine learning algorithms in use today. They are used in a variety of contexts for both regression and classification tasks. The advantage of decision trees over other classification algorithms is its intuitive nature making it easier to understand as well as being a white box algorithm allowing the practitioner to see how decision are being made.

In this paper, we consider using decision trees and random forests to determine the age of Abalone. Originally, the number of rings (which are subsequently used to determine the age) in Abalone dataset are recorded as an integer and as such regression techniques can be used to determine the rings/age. Instead, we re-cast the problem into a classification task by grouping the rings into 4 distinct groups and then use the CART implementation of decision trees and random forests to determine the age group given the physical attributes of Abalone.

An investigation into how regularizing the decision tree affects its performance is carried out. For this, we employ cross validation grid search of a range of different hyperparameters which are used to regularize decision trees. The best hyperparameter combination is then used to fit the regularized decision tree. The results of this method of regularizing are compared to a post-pruning technique known as cost-complexity pruning. Random forests with different sizes are also investigated to determine whether they provide superior results than tunes decision trees. In each case, repeated experiments are carried out to determine the mean and standard deviations which are used to make observations on the bias and variance trade-off at play.

In section II we present the literature review undertaken on the different approaches to Abalone age prediction as well as evaluation metrics in a multi-class imbalanced data setting such as the one at hand. In section III, we detail the methodology including exploratory data analysis, data cleaning and modelling techniques and the results are discussed in section IV. Finally, in section V we note the future work that can be carried out to further improve upon our results.

II. Literature review

A. Approaches to Abalone Age Prediction

Numerous approaches have been undertaken in using the Abalone dataset to estimate the Abalone age over the past 2 decades. Some approaches treat this as a regression problem while others use the classification problem setting.

CLOUDS, which was one of the best decision tree algorithms in the 1990s, achieved an accuracy of 26.4% when implemented on the Abalone dataset. At the time, due to computational power limitations, this algorithm's main contribution was a computationally efficient method for estimating the split at every internal node to grow the tree [3].

More recently, an ordinary least squares regression approach has been used with different combinations of the features to predict the age of Abalone [4]. Like our observations, they have also noted the presence of

multicollinearity in the features. As such, they have experimented with incorporating log transforms and other power transforms of features in the model specification.

Neural network approaches have been applied using various topological variances in the network architectures, including convolutional approaches and residual neural network architectures, to achieve 79.09% accuracy [5].
Reference [6] have employed more advanced neural network variations to the Abalone dataset. They use the Cascade Correlation learning architecture which trains a neural network by dynamically altering its topology starting from only a fully connect input and output layer and adding hidden layers one by one (evaluated by correlation with the output layer). They suggest that this leads the hidden layer neurons to learn more different features from data and achieve 76.8% accuracy on the Abalone dataset. Like our approach, they also recast the problem as a classification problem, however they use 3 classes whereas we use 4 classes. Additionally, like us, they have also noted a strong overlap between different classes and suggest that the Abalone dataset may not have enough features to distinguish different ages.

As our approach is to use CART decision trees and random forests for Abalone age classification, a comprehensive review of the CART algorithm was undertaken [7]. Importantly, this paper suggests that the CART algorithm can automatically adjust to class imbalance in the training data. This ability of the CART algorithm enables us to use it for our problem. As decision trees are prone to overfitting, the cost-complexity pruning algorithm [8] was reviewed as a regularization technique. The implementation of this pruning technique in scikit-learn was also reviewed [9].

A review of random forest algorithms was undertaken with a view to understand how it introduces more randomness as compared to decision trees and the effect of this on the overall bias and variance of the predictions made using random forests. Reference [10] applies the random forest algorithm on many different datasets including the Abalone dataset, however it is in the regression setting and not classification.

B. *Evaluation Metrics*

In the 4-class adaptation of the Abalone dataset, we are presented with a multi-class problem with imbalanced class distributions. However, commonly used evaluation metrics such as accuracy, precision, recall, F1-score etc. were designed to cater for binary classification settings [11], not multi-class classification and nor were they intended to work where class imbalance is present [12]. Nevertheless, these metrics can be adapted for multi-class classification by using macro-averaging where the arithmetic mean of the metrics on each class is calculated to give a global metric. Balanced accuracy is one such metric which does not inflate the overall accuracy score due to the accuracy of a high frequency class dominating over other classes' accuracy. However, an implicit assumption of using an arithmetic mean is that all classes have the same importance which is not usually the case. Another type of averaging is the weighted mean of the partial metrics where the weights represent the prior distributions of the classes. Other methods which use class weighting schemes to account for differences in class importance were also reviewed and are discussed in [13].

III. Methodology

This section details the methodology; the results and the accompanying discussion in is in the section IV.

A. *Data Pre-processing and Visualizations*

1) *Data*

The original data has 4177 observations with nine attributes which are described in the following table:

TABLE I.

Name	Type	Description	Unit of measurement
Sex	Nominal	3 levels (male, female, infant)	M, F or I
Length	Continuous	Longest shell measurement	Millimetres
Diameter	Continuous	Perpendicular to length	Millimetres
Height	Continuous	Height with meat in shell	Millimetres
Whole weight	Continuous	Whole abalone weight	Grams
Shucked weight	Continuous	Weight of meat	Grams
Viscera weight	Continuous	Gut weight after bleeding	Grams
Shell weight	Continuous	Weight after being dried	Grams
Rings	integer	+1.5 gives age in years	n/a

There were no null/NaN values encountered.

2) *Pre-processing*

a) The following table gives the summary statistics of the data.

TABLE II.

Statistic	Length	Diametre	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177	4177	4177	4177	4177	4177	4177	4177
mean	0.524	0.408	0.14	0.829	0.359	0.181	0.239	9.934
std	0.12	0.099	0.042	0.49	0.222	0.11	0.139	3.224
min	0.075	0.055	0	0.002	0.001	0	0.002	1
25%	0.45	0.35	0.115	0.442	0.186	0.094	0.13	8
50%	0.545	0.425	0.14	0.8	0.336	0.171	0.234	9
75%	0.615	0.48	0.165	1.153	0.502	0.253	0.329	11
max	0.815	0.65	1.13	2.826	1.488	0.76	1.005	29

It was noted that the minimum value for 'Height' recorded is 0. The following two observations were found to have 'Height' recorded as 0.

TABLE III.

Sex	Length	Diametre	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
I	0.43	0.34	0	0.428	0.2065	0.086	0.115	8
I	0.315	0.23	0	0.134	0.0575	0.0285	0.3505	6

These observations have non-zero values for other attributes and thus it was reasonably concluded that the height for these observations was recorded incorrectly. These observations were removed at this stage of the exploratory data analysis resulting in 4175 observations.

b) The 'Sex' attribute was encoded using the following scheme:

- Sex 'M' encoded as 0
- Sex 'F' encoded as 1
- Sex 'I' encoded as -1

c) To recast the original regression problem into a classification problem, the 'Rings' attribute was encoded into 4 classes using the following scheme and subsequently re-named 'Age group' to reflect the attribute appropriately:

- Rings between 0 and 7 (inclusive) encoded as class 1
- Rings between 8 and 10 (inclusive) encoded as class 2
- Rings between 11 and 15 (inclusive) encoded as class 3
- Rings greater than or equal to 16 encoded as class 4

3) Exploratory Data Analysis and Visualizations

a) Pairplot

The pair plot given in appendix A was constructed to give a visual summary of the continuous type features (that is, all features except Sex) and the target. The kernel density estimate (KDE) plots on the diagonal show the features' distribution while the off-diagonal plots are scatter plots of two features at a time. The key observations made are as follows:

- Looking at the KDE plots, there is skew in all the continuous features with Length and Diameter exhibiting negative skew while the other features exhibit positive skew.
- All the feature pair scatterplots exhibit a roughly (positive) linear relationship with each other. This was confirmed by constructing a correlation matrix of the continuous features and the target; further discussed in the Correlation Matrix section.
- There is an apparent curvature in some feature pair scatterplots, for example, the scatterplot of Whole weight and Diameter. This indicates a possible logarithmic/higher order dependence between some features. No data transformations were made to alter this.
- There is an apparent 'funnel effect' in some feature pair scatterplots compared to others, for example, Shell weight and Shucked weight scatterplot exhibits the effect to a greater degree than the Whole weight and Shucked weight scatterplot. This indicates the presence of heteroskedasticity in the data. No data transformations were made to alter this.
- All the scatterplots with Height as a feature clearly show 2 outlier values. This was confirmed by constructing the boxplot of height; further discussed in the Boxplot section.
- It is especially noteworthy that the scatterplot of the target classes exhibits a high degree of overlap between the four classes across all the continuous features. This was noted as it could cause difficulties for the algorithms when finding suitable (discriminative) decision boundaries.

To better visualize how the different target classes constitute the feature space, the pair plot in appendix B was constructed for the continuous features and colour coded by the Age group.

- The KDE plots now show the distributions of the different target classes for each feature. The general observation of skew being present is retained at the target class level as well. Length and Diameter KDEs exhibit negative skews for all the target classes while the other features exhibit positive skew.
- The KDE plots also show a lot of overlap in the individual feature densities for the different age groups.
- All the pair scatterplots have age group 1 data points dominate the lower left corner of the plot. However, the mid to high ranges of the plots have a lot of overlap in the age group composition. For example, in the Shell weight and Diameter pair scatterplot, while age group 1 is distinctly visible in the lower left corner of the plot, the other age groups' distinctions are less clear to detect (visually) due to a high degree of overlap. As noted above, this could cause difficulties for the algorithms when finding suitable (discriminative) decision boundaries.

b) Correlation Matrix

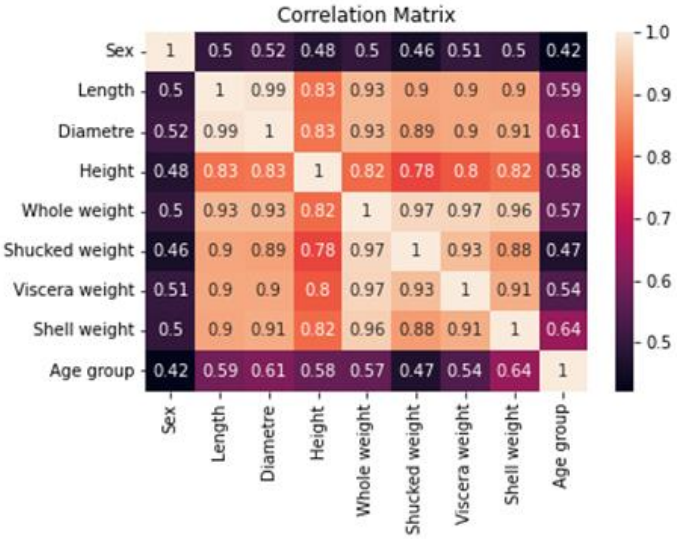


Fig. 1. Correlation Matrix of Abalone Features

- All the continuous features exhibit high positive correlations with each other as visible in the scatterplots. This is expected as the continuous feature types represent physical measurements of Abalone which are likely to be positively correlated. For example, if the diameter increases, the height is also likely to increase and so is weight.
- Although, the high within (continuous) feature correlations are expected, they present the multi-collinearity problem.

c) Boxplot of Height

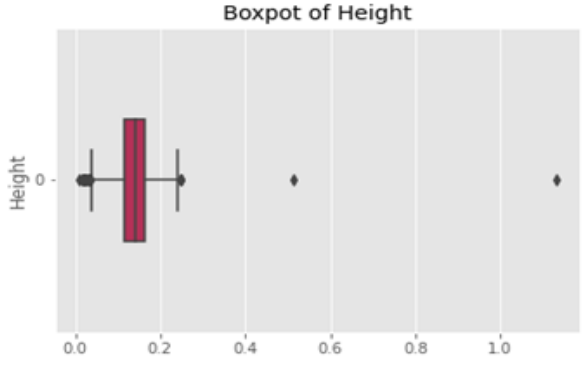


Fig. 2. Boxplot of the Height feature

- The boxplot for height shows the 2 outlier values around 0.5 mm and 1.3 mm. These observations were removed at this stage of the exploratory data analysis resulting in 4173 observations.

d) Histogram of Sex

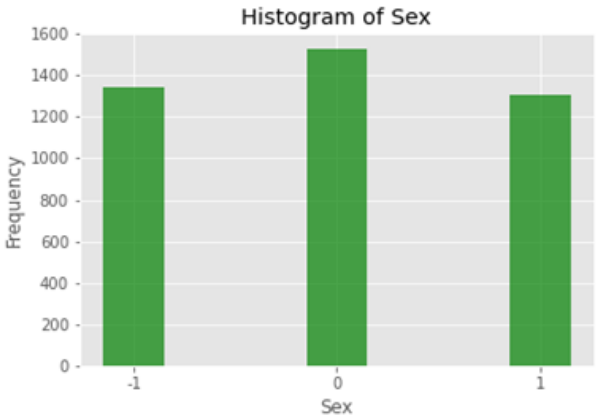


Fig. 3. Histogram of Sex feature

- The histogram exhibits roughly equal distribution of Sex.

The following table shows the number of observations:

TABLE IV.

Sex	Number of Observations
-1 (Infant)	1340
0 (Male)	1527
1 (Female)	1306
Total	4173

e) Histogram of Target Class

- The histogram exhibits imbalance in the target class distribution.

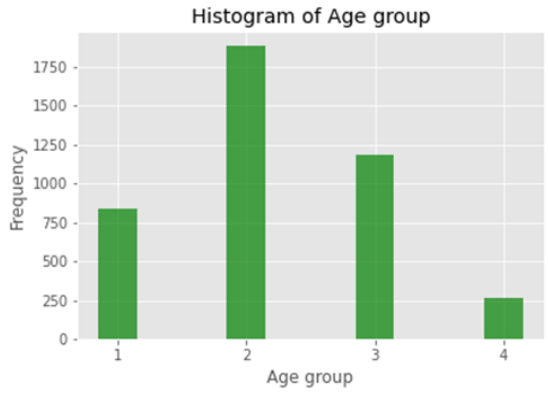


Fig. 4. Target class distribution

The following table shows the number of observations in each class:

TABLE V.

Class	Number of Observations
1	838
2	1888
3	1186
4	261
Total	4173

B) Data Preparation

As noted above, based on the exploratory data analysis and visualizations, we removed 4 observations from the dataset, 2 of which were observations with height recorded as '0' hence inaccurate observations and 2 which were outliers for the height feature.

The resulting 4173 observations were split into 60% training set and 40% testing set using stratified sampling to maintain the ratio of class distributions in train and test set. In subsequent modelling the same split of data was used throughout.

C) Evaluation Metrics

Our problem is a multi-class setting with an imbalanced data set. As noted in the literature review, balanced accuracy is a global metric which does not inflate the overall accuracy score due to the accuracy of a high frequency class dominating over other classes' accuracy. In our case, the assumption of all classes having the same importance is not necessarily reasonable. As such, the weighted average accuracy has been adopted as the main evaluation tool for our models. Other partial metrics such as each classes' precision and recall are also given.

In the multi-class setting, the accuracy of each class can be calculated as the number of instances of class j correctly classified divided by the total number of observations for class j. Then, the formula for the accuracy score of each class is:

$$Acc = \frac{TP}{TP + FN} \quad (1)$$

Where TP (true positives) and FN (false negatives) are of that class.

Note that this is also the recall score of each class. As such, in multi-class classification, the weighted average of each classes' recall score can be used as the global metric for accuracy. The weights reflect the prior distribution of the classes.

The formula for weighted-average accuracy score is:

$$\text{Weighted} - \text{Average Accuracy} = \sum_{j=1}^{K} \frac{C_j}{N} Acc_j \quad (2)$$

Where N denotes the total number of observations, K is total number of classes and Cj is the number of class j observations.

In scikit-learn, when using the usual "accuracy_score" metric on multi-class classification data, the score is calculated as above, it is the weighted average of the recall scores for each class.

D) Modelling

We now detail how the modelling was undertaken. The results are presented in the next section.

1) CART Decision Tree Without Regularization

We use the (optimised version) of the CART algorithm implementation in Python scikit-learn library [9].

The key hyper-parameters in scikit-learn used to regularize the decision tree and avoid overfitting are:

- Max_depth: maximum depth of the decision tree
- Max_leaf_nodes: maximum number of leaf nodes
- Min_samples_leaf: minimum number of samples in a leaf node
- Min_samples_split: minimum number of samples per node, before it can be split

Firstly, 10 experiments were conducted to fit decision trees on the same training and testing data without any regularization. The train and test weighted-accuracy scores on each experiment were collected along with the precision and recall scores for each class. These scores as well as the mean score and confidence interval are given in Table VI and Table VII.

The confidence intervals have been calculated using the following formula:

$$\bar{x} \pm t_{n-1} \frac{s}{\sqrt{n}} \tag{3}$$

Where n=10, $\bar{x}$ = mean of the metric, s is the sample standard deviation of the metric and t_{n-1} is the 97.5% critical value of the t-distribution with n-1 degrees of freedom. In our case, we have 10 samples and as such the critical value is calculated using 9 degrees of freedom yielding t_{n-1} = 2.262. The t-distribution is used as the population standard deviation is not known.

2) CART Decision Tree with Regularization

Grid search with five-fold cross validation was used on the training data to test different hyperparameter combinations to regularize the fitted decision tree. The grid used the following hyperparameter combinations:

- Max_depth: [3,4,5,6,7,8,9,10,11,12,13,14]
- Max_leaf_nodes: [50,100,150,200,250,300,350,400,450]
- Min_samples_leaf: [5,15,25,35,45,55]
- Min_samples_split: [2,5,15,25,35,45,55,65]

As such, a total of 12x9x6x8 = 5184 combinations were tested.

The weighted-accuracy evaluation metric was used in the grid search to select the best hyperparameters, these are given in Table VIII.

Using the best hyperparameters, the decision tree was fit on the whole training set and tested on the testing set. Again, 10 experiments were conducted and the train and test weighted-accuracy scores on each experiment were collected. Additionally, precision and recall scores for each class on each of the 10 experimental runs were also collected. These scores as well as the mean score and confidence interval are given in Table IX and Table X.

3) Cost-Complexity Pruning

Using the hyperparameters shown above is one way to fit and regularize a decision tree by early stopping. Another approach is to let the decision tree grow unrestricted and then apply post-pruning, the algorithm we use is the cost complexity pruning algorithm.

A decision tree will attempt to minimize the misclassification rate by continuing to grow. This algorithm uses a cost function which constitutes the misclassification rate of the decision tree and an additional complexity term. The complexity is defined as the number of leaf nodes in the tree [14]. The modified cost function is parametrized by a single parameter α as follows:

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}| \tag{4}$$

where $|\tilde{T}|$ is the number of leaf nodes in the tree T, R(T) is the total misclassification rate of the leaf nodes and α is the penalty imposed on each leaf node [9]. The α parameter controls the trade-off between fit and complexity, in other words, bias and variance.

For a given value of α, the subtree which minimizes this cost function is the final pruned tree. Using α=0 is equivalent to no regularization and no pruning whereas α=1 will prune away the whole tree leaving only a root node [14]. This is because the splits at the bottom of the tree that reduce R(T) the least will be pruned away [9]. In the implementation of this algorithm, α is incrementally increased to a value where all splits are pruned. The most suitable of these incremental α's is then chosen by testing on test set or through cross-validation.

We use scikit-learn to implement cost-complexity pruning though the "cost_complexity_pruning_path" functionality [15]. The α which maximises the testing set accuracy is chosen. The decision tree is then fit using this value of α and no other early stopping regularization hyperparameters are used. The train and test weighted-accuracy scores on each experiment were collected along with the precision and recall scores for each class. These scores as well as the mean score and confidence interval are given in Table XI and Table XII.

4) Random Forests

Lastly, an ensemble approach using Random Forests was also used. Random Forests build numerous individual trees and then combine the predictions of the individual trees (through majority voting) to arrive at the final class prediction in the classification setting. Whereas decision trees try to find the feature and threshold value which results in the maximum information gain at each split, random forests introduce extra randomness by searching for the best feature in a random subset of the features [16].

Additionally, they use the bagging approach and only use data from a random sample of the training data to train each tree. These sources of randomness enable the random forest algorithm to build diverse uncorrelated trees and generally get a better overall model than individual decision trees achieving lower variance but higher bias [17].

In addition to the hyperparameters used in fitting decision trees, random forests have an extra hyperparameter for the number of decision trees to use in the ensemble. In our

approach, we investigate using different number of decision trees in the ensemble however each time restricting the maximum depth of the trees to 5; these form the different models. We conduct 10 experiments on each of these models and collect the weighted-accuracy scores. Mean weighted-accuracy scores and confidence intervals are provided for each of the models in Table XIII.

IV. Results

A. CART Decision Tree Without Regularization

TABLE VI

Experiment	Train Accuracy	Test Accuracy	Class 1 Precision	Class 1 Recall	Class 2 Precision	Class 2 Recall	Class 3 Precision	Class 3 Recall	Class 4 Precision	Class 4 Recall
1	1	0.534	0.653	0.69	0.583	0.549	0.436	0.459	0.269	0.269
2	1	0.539	0.669	0.699	0.583	0.556	0.443	0.455	0.27	0.288
3	1	0.548	0.66	0.696	0.594	0.563	0.453	0.476	0.297	0.288
4	1	0.533	0.66	0.69	0.579	0.55	0.422	0.446	0.31	0.298
5	1	0.541	0.646	0.681	0.587	0.556	0.453	0.472	0.292	0.298
6	1	0.534	0.653	0.69	0.583	0.549	0.436	0.459	0.269	0.269
7	1	0.546	0.673	0.69	0.591	0.57	0.442	0.469	0.28	0.25
8	1	0.533	0.653	0.681	0.581	0.553	0.44	0.463	0.238	0.231
9	1	0.541	0.678	0.687	0.586	0.567	0.436	0.453	0.274	0.279
10	1	0.531	0.668	0.69	0.577	0.549	0.429	0.448	0.259	0.269

Table VI presents the results for the 10 experiments conducted to fit decision trees without any regularization.

TABLE VII

			95% CI	
Metric	Mean	Std	Lower Limit	Upper Limit
Train Accuracy	1	0	1	1
Test Accuracy	0.5379	0.0058	0.5338	0.5420
Class 1 Precision	0.6613	0.0103	0.6539	0.6687
Class 1 Recall	0.6894	0.0056	0.6854	0.6934
Class 2 Precision	0.5844	0.0052	0.5807	0.5881
Class 2 Recall	0.5562	0.0079	0.5506	0.5618
Class 3 Precision	0.4390	0.0096	0.4321	0.4459
Class 3 Recall	0.4600	0.0100	0.4528	0.4672
Class 4 Precision	0.2758	0.0204	0.2612	0.2904
Class 4 Recall	0.2739	0.0213	0.2587	0.2891

Table VII shows the mean, standard deviation and the 95% confidence interval for the train and test weighted-accuracy scores as well as the precision and recall metrics for the 4 classes.

It can be seen that the the mean training weighted-accuracy score is 1 and the standard deviation is 0. On the other hand, the mean test weighted-accuracy score is much lower and the standard deviation is higher than 0. This large difference indicates that the decision trees built in these 10 experiments are all overfitting the data. This was expected as we noted above that decision trees without regularization tend to overfit the training data.

While this model will have no bias, it will have high variability reflected in the higher standard deviation for the test weighted-accuracy scores compared to the standard deviation on the training weighted-accuracy. As such, the models' performance on the test data is quite low (just better than random guessing).

It is also noteworthy that the mean precision and recall scores (recall is equivalent to accuracy for a single class for multi-class settings, as noted above) on the testing set for class 1 are higher than other classes' while these scores are the lowest for class 4. In the discussion on the pairplots (in the visualization section), we noted that the overlap between class 1 and other classes is less compared to the overlap between other classes. This could be making it easier for the classifier to build discriminative decision boundaries to classify class 1 data better than other classes' data and thus attaining higher precision and recall scores. Another contributing factor is the difference in class distributions, where class 4 has the lowest number of observations.

B. CART Decision Tree With Regularization

GridSearch with 5-fold cross validation was used to find the best hyperparamater combination. The best combination are given in Table VIII.

TABLE VIII

Hyperparameter	Selected Combination
Maximum depth	8
Maximum leaf nodes	50
Minimum samples leaf	35
Minimum samples split	5

The highest cross validation weighted-accuracy score achieved with these hyperparamters was 0.63. These hyperparamatres enforce early stopping and thus prevent the decision tree from overfitting as it did previously with no regularization.

Table IX shows the results for the 10 experiments conducted to fit decision trees using the regularization parameters in Table VIII.

TABLE IX

Experiment	Train Accuracy	Test Accuracy	Class 1 Precision	Class 1 Recall	Class 2 Precision	Class 2 Recall	Class 3 Precision	Class 3 Recall	Class 4 Precision	Class 4 Recall
1	0.663	0.635	0.810	0.764	0.627	0.718	0.531	0.531	0.769	0.096
2	0.663	0.635	0.810	0.764	0.627	0.718	0.531	0.531	0.769	0.096
3	0.663	0.635	0.810	0.764	0.627	0.718	0.531	0.531	0.769	0.096
4	0.663	0.635	0.810	0.764	0.627	0.718	0.531	0.531	0.769	0.096
5	0.663	0.635	0.810	0.764	0.627	0.718	0.531	0.531	0.769	0.096
6	0.663	0.635	0.810	0.764	0.627	0.718	0.531	0.531	0.769	0.096
7	0.663	0.635	0.810	0.764	0.627	0.718	0.531	0.531	0.769	0.096
8	0.663	0.635	0.810	0.764	0.627	0.718	0.531	0.531	0.769	0.096
9	0.663	0.635	0.810	0.764	0.627	0.718	0.531	0.531	0.769	0.096
10	0.663	0.635	0.810	0.764	0.627	0.718	0.531	0.531	0.769	0.096

In this case, the decision tree produced the same scores for all metrics for all 10 experiments.

TABLE X

			95% CI	
Metric	Mean	Std	Lower Limit	Upper Limit
Train Accuracy	0.663	0	0.663	0.663
Test Accuracy	0.635	0	0.635	0.635
Class 1 Precision	0.810	0	0.810	0.810
Class 1 Recall	0.764	0	0.764	0.764
Class 2 Precision	0.627	0	0.627	0.627
Class 2 Recall	0.718	0	0.718	0.718
Class 3 Precision	0.531	0	0.531	0.531
Class 3 Recall	0.531	0	0.531	0.531
Class 4 Precision	0.769	0	0.769	0.769
Class 4 Recall	0.096	0	0.096	0.096

Table X shows the mean, standard deviation and the 95% confidence interval for the train and test weighted-accuracy scores as well as the precision and recall metrics for the 4 classes.

There is an increase of roughly 0.10 in the mean testing weighted-accuracy scores as compared to the results of the unregularized tree, however the mean score for the training set has decreased significantly from 1 to 0.663. The closeness of the training and testing weighted accuracy score suggests that the tree maybe underfitting the data even though grid search was used to find the hyperparameter combination yielding maximal test weighted-accuracy score.

Interestingly, we also notice that the precision rates for all classes have increased with the regularized decision tree, the largest increase coming from class 4. Also, recall rates have increased for all classes except class 4 which has decreased substantially. This indicates that when we enforce early stopping, the tree tries to fit better on classes with larger proportion of samples and thus does poorly with recall for class 4 instances. The trade-off between precision and recall is also contributing to this.

In particular, a visualization of the fitted decision tree on the 10th experimental run is provided in Appendix C. As this tree is too large, another visualization of the same tree but with lesser depth is provided in Appendix D. The area marked in red on the tree in Appendix D is zoomed in an provided in Appendix E along with its translation into "if" and "then" rules.

C) Cost-Complexity Pruning

Cost complexity pruning was applied on the training set using the "cost_complexity_pruning_path" functionality in scikit-learn. As described above, more and more of the decision tree is pruned as the alpha value increases. This results in higher impurity in the decision tree's leaves over time. This relationship is depicted in Figure 5. The metric used for impurity was the gini index.

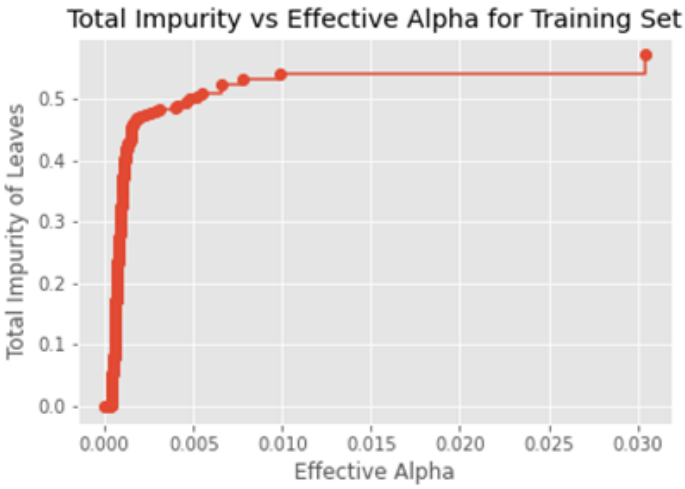


Fig. 5. Total Impurity vs Effective Alpha

We notice that the impurity quickly rises with an increase in α and then begins to level off beyond $\alpha > 0.005$.

Alternative ways to see the effect of increasing α is by looking at its impact on the number of nodes in the tree and tree depth. These relationships are depicted in Figures 6 and 7 respectively.

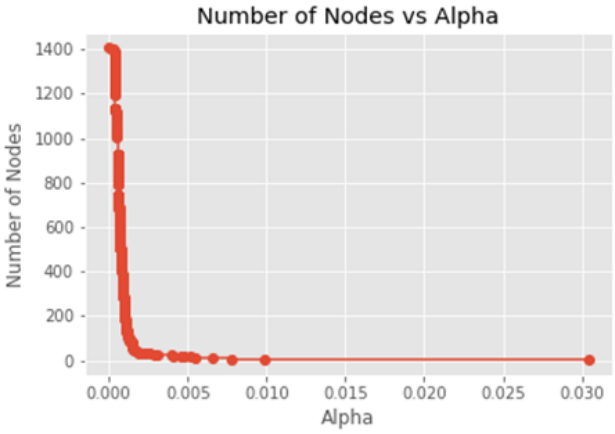


Fig. 6. Number of Nodes vs Effective Alpha

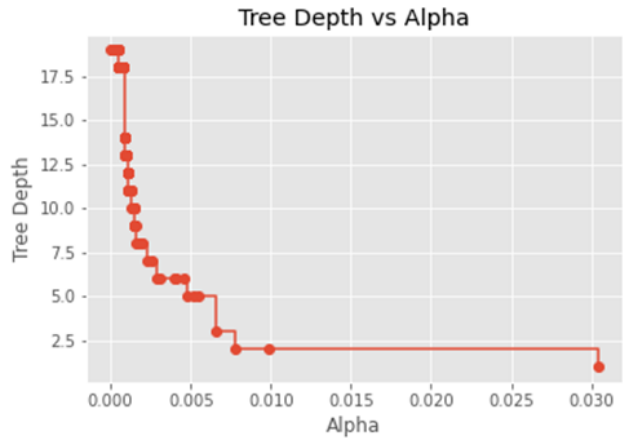


Fig. 7. Tree Depth vs Effective Alpha

A sharp initial decline in the number of nodes and depth in the decision tree is clearly visible. Similar to Figure 5, this starts to level off beyond $\alpha > 0.005$. These trends are related since the tree depth directly impacts the number of nodes in the tree which impacts the total impurity in the leaves.

As described in the methodology section of the paper, the cost-complexity pruning implementation obtains incremental values of α ranging from 0 to 1 and the best α is chosen based on performance on the test set. The results of the different α values on the training and test set weighted-accuracy scores are presented in Figure 8.

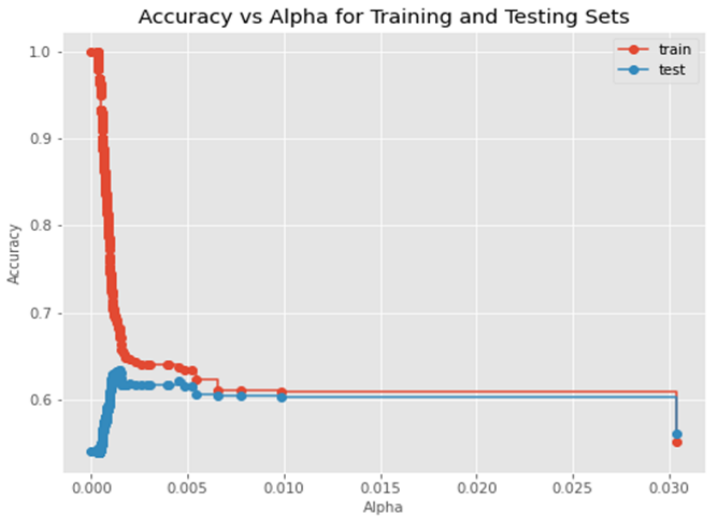


Fig. 8. Train and Test Weighted-Accuracy vs Effective Alpha

The highest value of the weighted-accuracy score is 0.635 obtained by $\alpha = 0.001$. However, like the other figures presented above, we notice a significant decline of the weighted-accuracy in the training set while the test set weighted-accuracy only increases by about 0.1. This indicates that we are trading a lot of bias for little gain in the generalization power/variance reduction of the fitted model. This was also noted when we applied early stopping regularization above. The reduction in the training weighted accuracy was larger than the gain in testing weighted accuracy.

Ten experiments were conducted to fit decision trees using the best α obtained from cost complexity pruning and no other 'early stopping' regularization parameters. The results are presented in table XI.

Table XI.

Experiment	Train Accuracy	Test Accuracy	Class 1 Precision	Class 1 Recall	Class 2 Precision	Class 2 Recall	Class 3 Precision	Class 3 Recall	Class 4 Precision	Class 4 Recall
1	0.675	0.635	0.803	0.767	0.647	0.700	0.522	0.560	0.409	0.087
2	0.675	0.634	0.796	0.767	0.647	0.700	0.521	0.556	0.409	0.087
3	0.675	0.635	0.796	0.767	0.648	0.701	0.521	0.556	0.409	0.087
4	0.675	0.634	0.796	0.767	0.647	0.700	0.521	0.556	0.409	0.087
5	0.675	0.635	0.796	0.767	0.648	0.701	0.521	0.556	0.409	0.087
6	0.675	0.636	0.803	0.767	0.648	0.701	0.522	0.560	0.409	0.087
7	0.675	0.635	0.796	0.767	0.648	0.701	0.521	0.556	0.409	0.087
8	0.675	0.634	0.796	0.767	0.647	0.700	0.521	0.556	0.409	0.087
9	0.675	0.635	0.803	0.767	0.647	0.700	0.522	0.560	0.409	0.087
10	0.675	0.634	0.796	0.767	0.647	0.700	0.521	0.556	0.409	0.087

Table XII

			95% CI	
Metric	Mean	Std	Lower Limit	Upper Limit
Train Accuracy	0.675	0.000	0.675	0.675
Test Accuracy	0.635	0.001	0.634	0.635
Class 1 Precision	0.798	0.003	0.796	0.801
Class 1 Recall	0.767	0.000	0.767	0.767
Class 2 Precision	0.647	0.001	0.647	0.648
Class 2 Recall	0.700	0.001	0.700	0.701
Class 3 Precision	0.521	0.000	0.521	0.522
Class 3 Recall	0.557	0.002	0.556	0.559
Class 4 Precision	0.409	0.000	0.409	0.409
Class 4 Recall	0.087	0.000	0.087	0.087

As compared to the results with early stopping, the mean training weighted-accuracy has increased but mean testing weighted-accuracy of 63.5% is the same as the early stopping hyperparameters achieved. As such, we can conclude that using early stopping hyperparameters and using the cost complexity pruning (post pruning) do not have a marked difference on the weighted-accuracy.

However, it is noteworthy that there has been a sharp fall in the mean precision for class 4 while the recall has remained almost the same.

A visualization of the fitted decision tree on the 10^{th} experimental run is provided in Appendix F (depth has been limited due to size).

D) Random Forests

Including different number of trees in the random forest was investigated and ten experiments were carried out for each forest to obtain the mean, standard deviation and the confidence intervals as shown in table XIII.

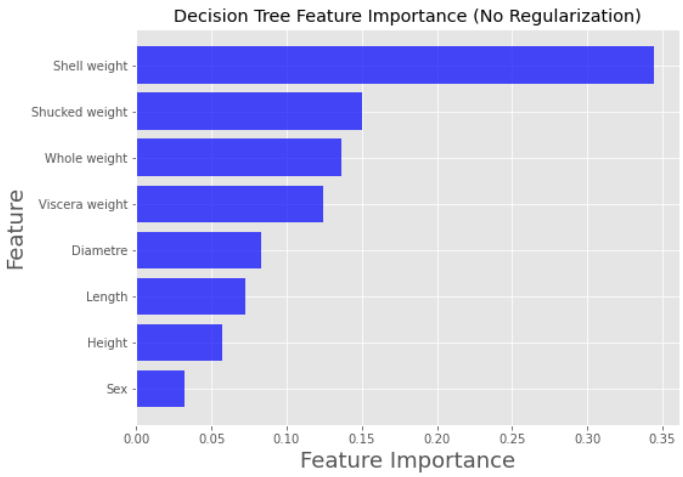
As discussed in the methodology section for random forests, the algorithm aims to build diverse trees to achieve lower variance but higher bias. The mean training weighted-accuracy is lower for all number of trees in the random forest classifier as compared to with a single decision tree using early stopping or cost complexity pruning. This points to the random forest model having higher bias. The testing weighted-accuracy standard deviation can be seen to be decreasing as the number of trees in the ensemble increases indicating the decrease in variance as the random forest size (and thus diversity) grows. However, the random forest has lower mean testing weighted accuracy for all forest sizes as compared to the single decision trees fit using both early stopping and cost complexity pruning.

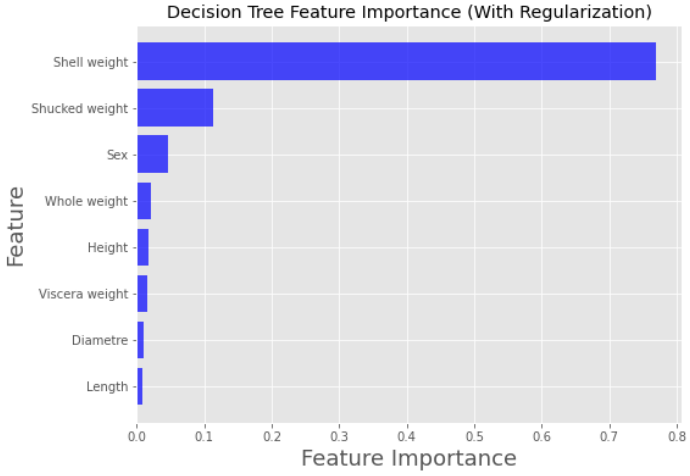
E) Feature Importance

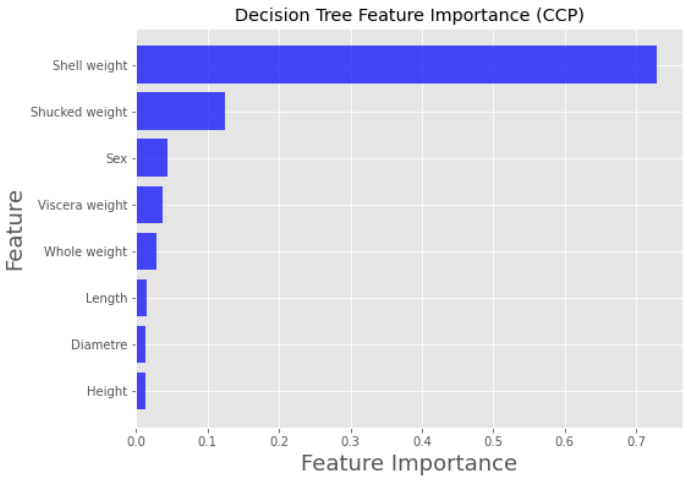
After a decision tree or random forest model is fit, its feature importances can be extracted to inform us of the features the algorithm thought were most useful in determining the decision boundaries. The bar plots of the feature importance from all 4 models are given in Figure 9.

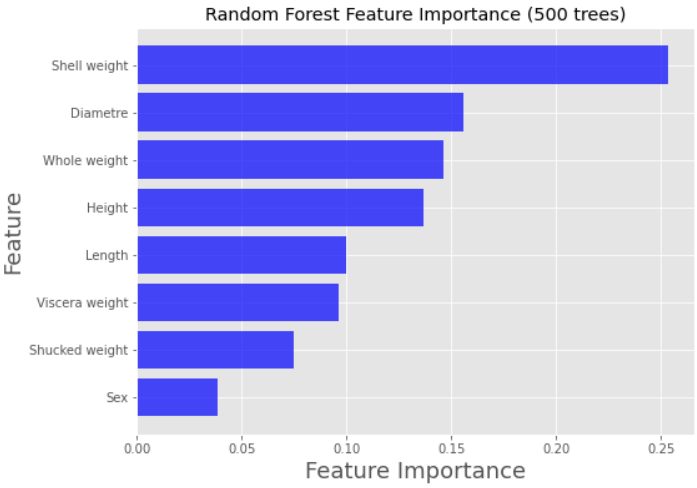
Table XIII

			95% CI				95% CI	
Number of Trees	Mean Train Accuracy	Std Train Accuracy	Lower Limit	Upper Limit	Mean Test Accuracy	Std Test Accuracy	Lower Limit	Upper Limit
20	0.649	0.005	0.645	0.653	0.617	0.004	0.614	0.620
40	0.650	0.002	0.649	0.651	0.619	0.003	0.617	0.621
60	0.651	0.003	0.649	0.653	0.616	0.003	0.614	0.618
80	0.650	0.002	0.649	0.651	0.617	0.003	0.615	0.619
100	0.652	0.003	0.650	0.654	0.618	0.002	0.617	0.619
200	0.652	0.001	0.651	0.653	0.620	0.002	0.619	0.621
300	0.652	0.002	0.651	0.653	0.619	0.002	0.618	0.620
400	0.651	0.001	0.650	0.652	0.620	0.002	0.619	0.621
500	0.651	0.002	0.650	0.652	0.619	0.002	0.618	0.620









It can be seen that all four models gave the highest importance to the Shell weight feature meaning that this feature was the most useful in distinguishing between classes.

The decision tree fit without regularization gives higher importance scores to features other than Shell weight and thus reduces the score for shell weight (the scores are normalised). This is because there is no restriction, and it is allowed to grow fully, thus using other features to distinguish between classes. However, when the decision tree is regularized, it is effectively forced to learn the decision boundaries with less degrees of freedom resulting in higher importance for feature(s) that allow it to distinguish between classes with this constraint.

The feature importance scores with early stopping regularization and cost-complexity pruning are almost identical.

V. Future Work

We were able to achieve weighted-accuracy scores of ~64% by applying regularized decision trees and random forests. Future work could incorporate boosting techniques such as AdaBoost, Gradient Boosting and XGBoost to investigate whether they can further improve the performance.

We have noted class imbalance issues which reflected in the low precision and recall for class 4 as compared to other classes. Techniques such as random under sampling of the majority class combined with oversampling of the minority class, e.g., SMOTE can be used to obtain a more balanced dataset [18]. The same modelling technique described in this paper can then be applied on the balanced dataset to investigate the improvements in weighted-accuracy scores.

Lastly, we noted overlap between the target classes limiting the performance of the models. Including additional features such as environmental factors into the modelling can be investigated as these might introduce more separation amongst the classes.

VI. Conclusion

We investigated using decision tree and random forest models to predict the age class of Abalone. Regularization techniques such as early stopping, and cost-complexity pruning were employed to improve the generalisation of these models. We found that applying decision trees with regularisation improved our performance compared to without regularisation. However, we observed large increase in bias (drop in training accuracy) for a relatively smaller increase in predictive power of the model. This was the case with both early stopping and cost-complexity pruning. Moreover, both regularization techniques provided us similar performance on the testing data. Random forests of various sizes were also investigated, however, we found that although the variance decreased as the number of trees in the ensemble increased, the performance on the test set was lower than with decision trees.

REFERENCES

[1] K. Mehta, "Abalone Age Prediction Problem: A Review," International Journal of Computer Applications, vol 178,

no. 50, pp. 43-49, Sept. 2019. [Online]. Available:

https://www.ijcaonline.org/archives/volume178/number50/mehta-2019-ijca-919425.pdf

[2] UCI. "Abalone Data Set." UCI – Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/abalone (Accessed Nov. 3, 2021).

[3] K. Alsabti, S. Ranka and V. Singh, "CLOUDS: A Decision Tree Classifier for Large Datasets," *Electrical Engineering and Computer Science*, Paper 41, Aug. 1998. [Online]. Available: https://dl.acm.org/doi/10.5555/3000292.3000294

[4] M. M. Hossain and M. N. M. Chowdhury, "Econometric Ways to Estimate the Age and Price of Abalone," *Munich Personal RePEc Archive*, MPRA paper no. 89557, pp. 1-18, Jan. 2019. [Online]. Available:https://mpra.ub.unimuenchen.de/91210/1/MPRA_paper_91210.pdfM. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[5] E. Sahin, C. J. Saul, E. Ozsarfati and A. Yilmaz, "Abalone Life Phase Classification with Deep Learning," *International Conference on Soft Computing & Machine Intelligence (ISCMI)*, No. 5, Nov. 2018. doi: 10.1109/ISCMI.2018.8703232

[6] Z. Wang, "Abalone Age Prediction Employing A Cascade Network Algorithm and Conditional Generative Adversarial Networks," *Research School of Computer Science, Australian National University*, 2018.[Online].Available: http://users.cecs.anu.edu.au/~Tom.Gedeon/conf/ABCs2018/paper/ABCs2018_paper_35.pdf

[7] D. Steinberg, "Chapter 10 CART: Classification and Regression Trees," in *The top ten algorithms in data mining*, Taylor and Francis Group, 2009, ch 10, pp. 193-216.

[8] ML Wiki. "Cost-complexity Pruning." ML Wiki. http://mlwiki.org/index.php/Cost-Complexity_Pruning (Accessed Nov. 10, 2021).

[9] Scikit-learn Developers. "1.10. Decision Trees." Scikit-learn. https://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart. (Accessed Nov.11, 2021).

[10] L. Breiman, "Random Forests," Machine Learning, vol. 45, pp. 5-32, 2001, doi: https://doi.org/10.1023/A:1010933404324

[11] M. Hossin & M. N. Sulaiman, "A Review on Evaluation Metrics for Data Classification Evaluations," IJDKP, vol. 5, No. 2, pp. 1-11, March 2015, doi: 10.5121/ijdkp.2015.5201

[12] N. Japkowicz, "Assessment Metrics for Imbalanced Learning," in *Imbalanced Learning: Foundations, Algorithms, and Applications*, John Wiley &Sons, 2013, ch. 8, pp. 187-206.

[13] A. Gupta, N. Tatbul, R. Marcus, S. Zhou, I. Lee and J. Gottschlich, "Class-weighted evaluation metrics for imbalanced data classification," 2020, [Online]. Available: https://arxiv.org/pdf/2010.05995.pdf

[14] The Pennsylvania State University. "11.8.2 – Minimal Cost-Complexity Pruning."PennState.https://online.stat.psu.edu/stat508/lesson/11/11.8/11.8.2. (Accessed: Nov.10, 2021)

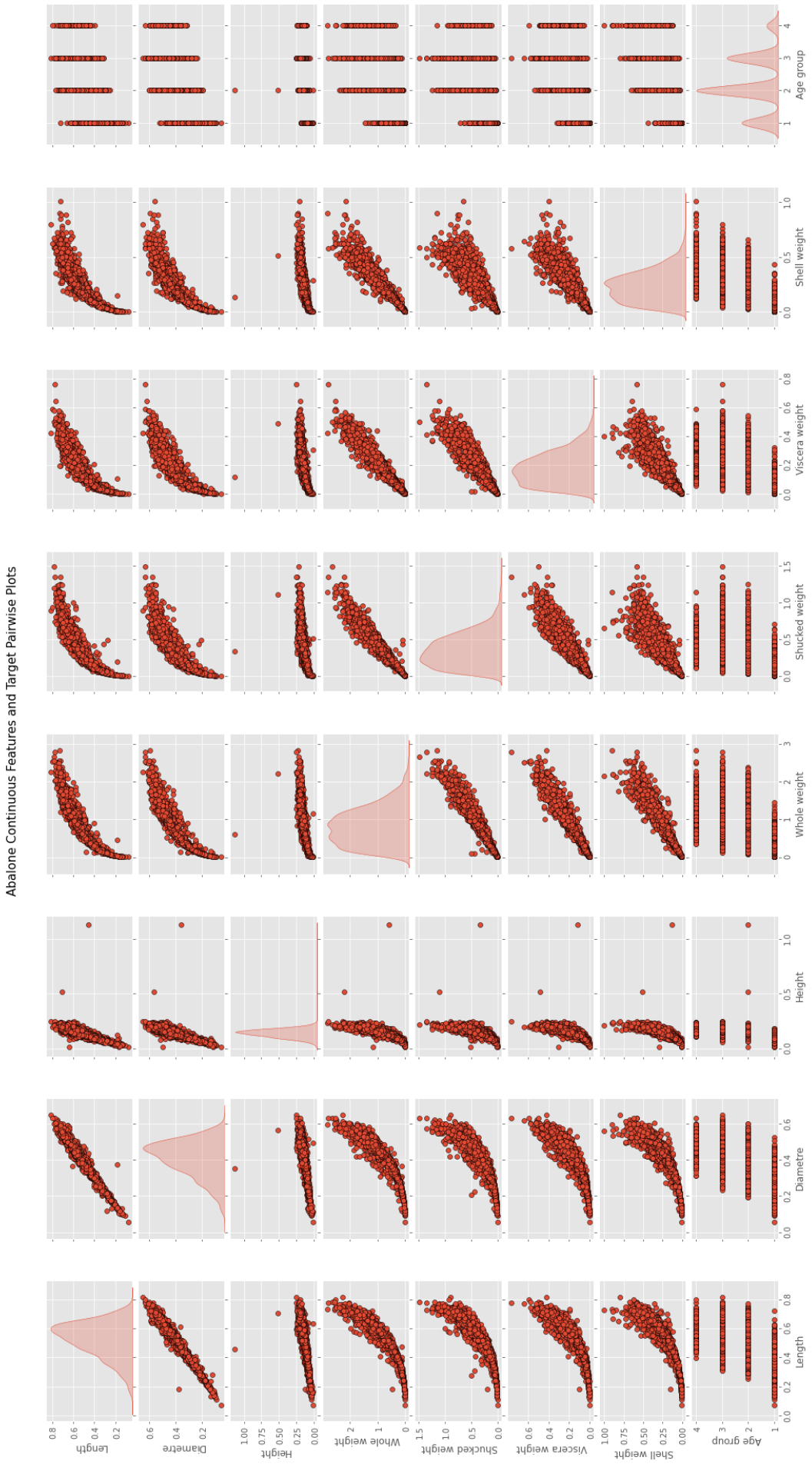
[15] Scikit-learn Developers. Post pruning decision trees with cost complexity pruning" Scikit-learn. 15https://scikitlearn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html. (Accessed Nov.13, 2021).

[16] L. Breiman, "Random Forests," Machine Learning, vol. 45, pp. 6-7, 2001, doi: https://doi.org/10.1023/A:1010933404324

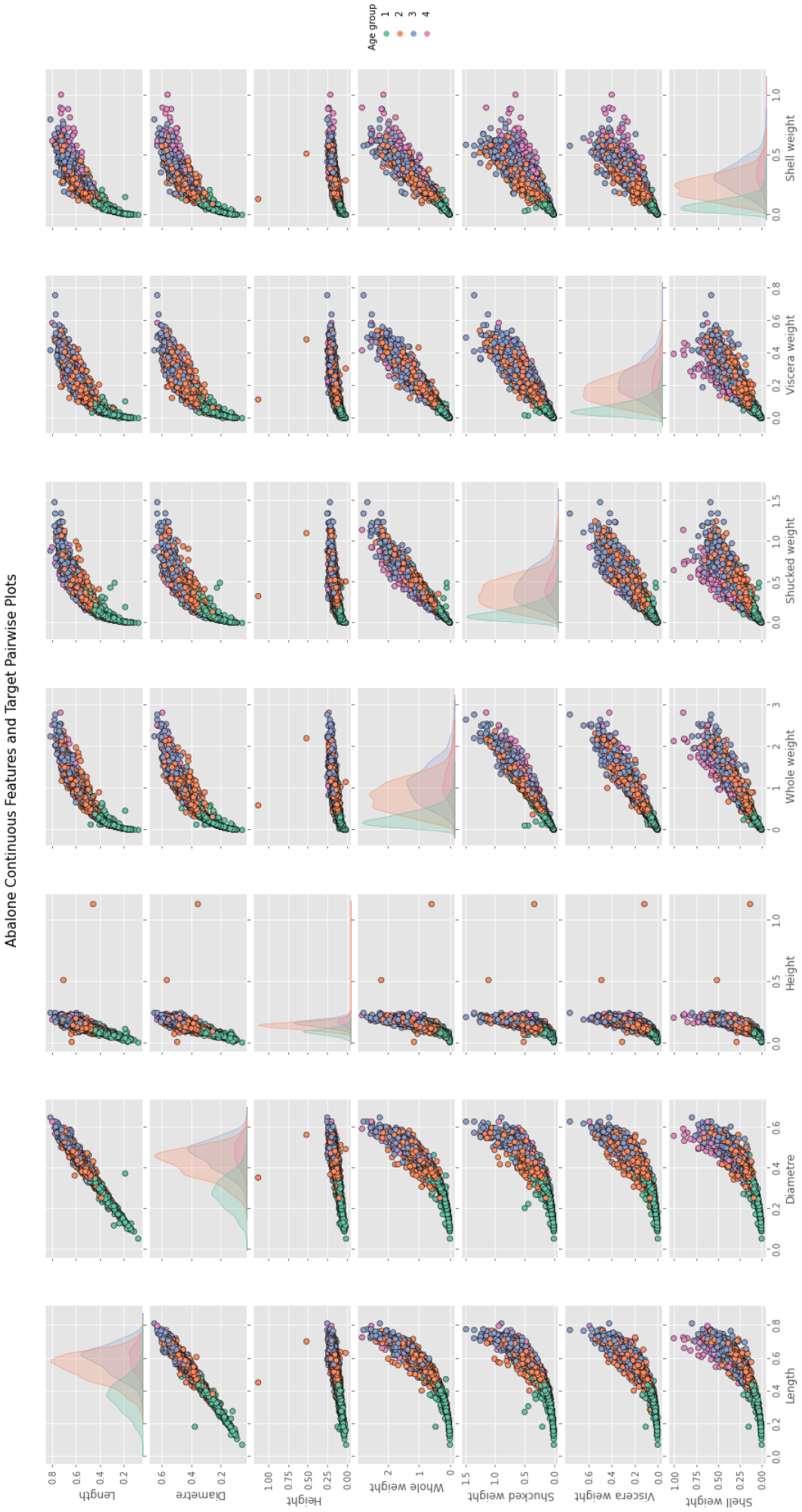
[17] A. Géron, Hands-on machine learning with Scikit-Learn and TensorFlow. 2nd ed. O'Reilly Media, Inc., 2019, p.197.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, & W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal Of Artificial Intelligence Research*, vol. 16, pp. 321-357, June 2002, doi: 10.1613/jair.953

Appendix A

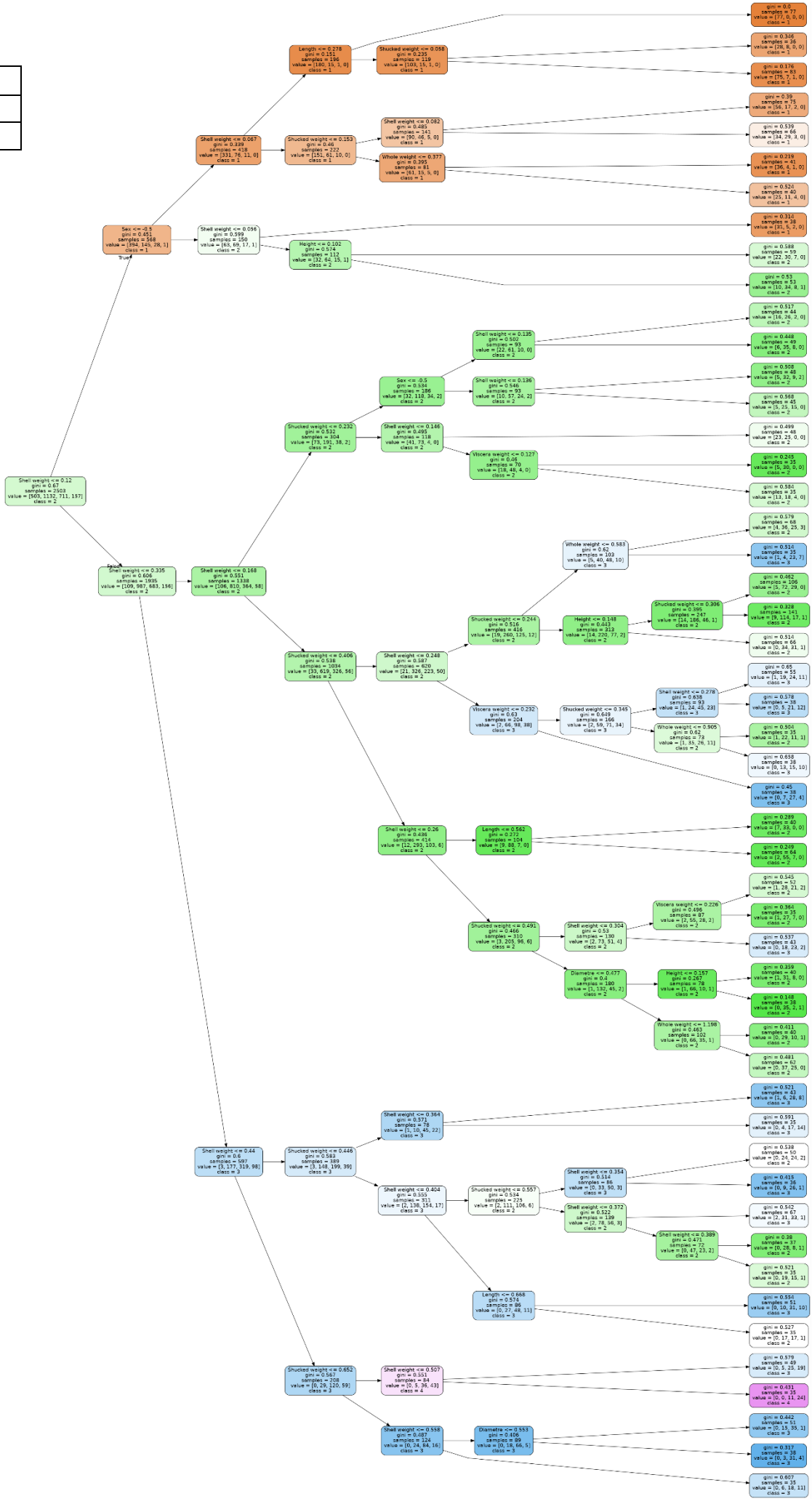

Abalone Continuous Features and Target Pairwise Plots
Length
Diameter
Height
Whole weight
Shucked weight
Viscera weight
Shell weight
Age group

Appendix B

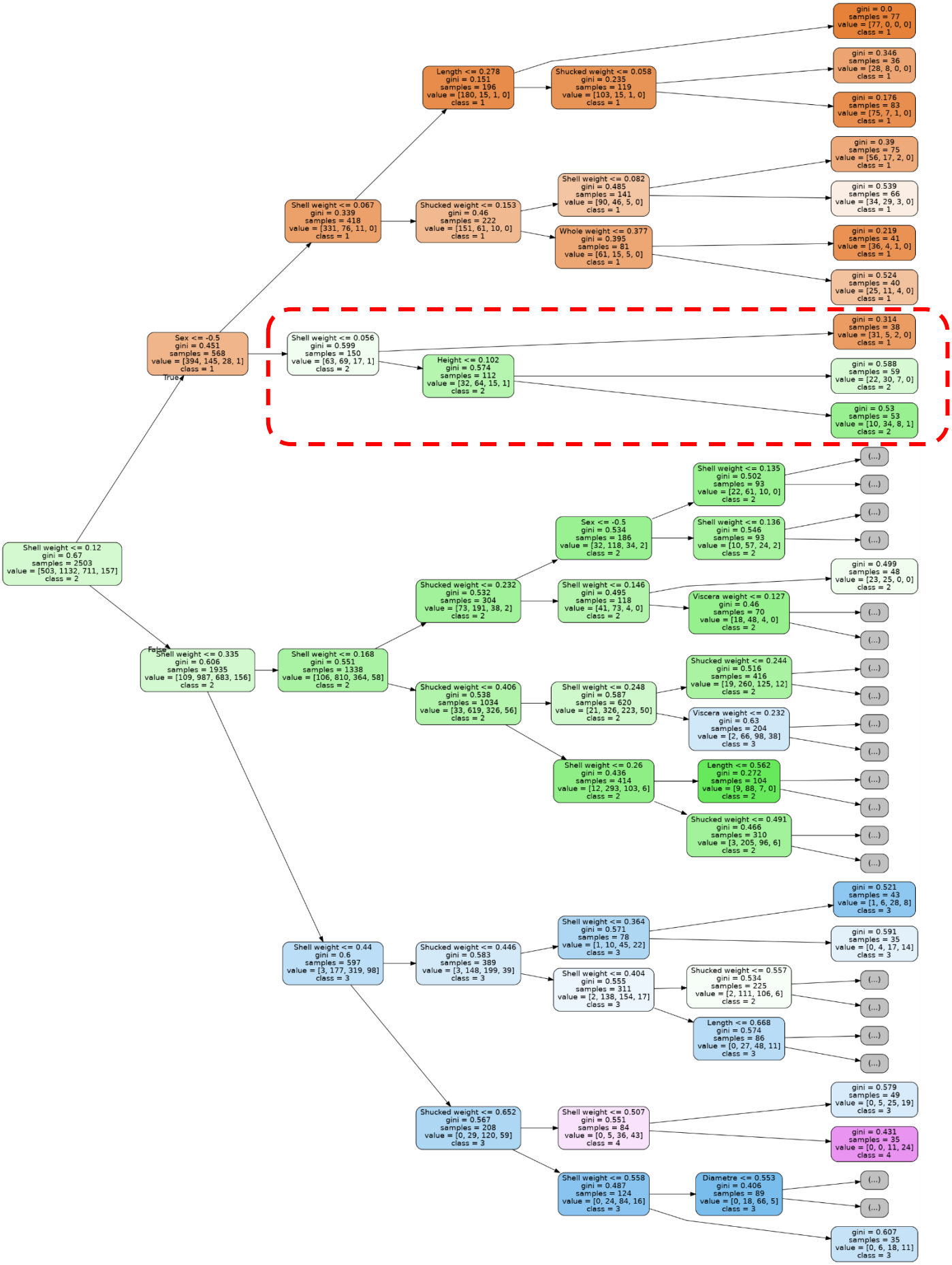

Abalone Continuous Features and Target Pairwise Plots
Age group
1
2
3
4
Length
Diametre
Height
Whole weight
Shucked weight
Viscera weight
Shell weight

Appendix C

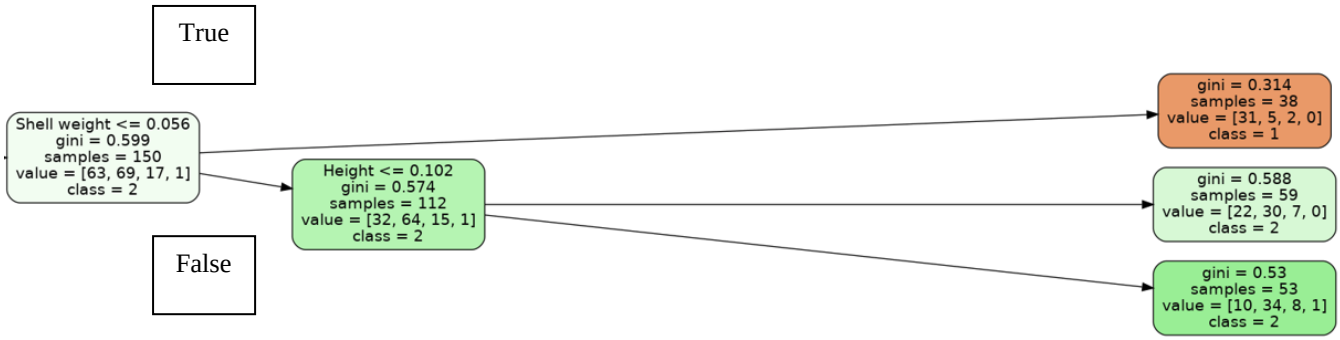
Depth	8
Nodes	99
Leaves	50



Appendix D


Shell weight <= 0.12
gini = 0.67
samples = 2503
value = [503, 1132, 711, 157]
class = 2
True
False
Sex <= -0.5
gini = 0.451
samples = 568
value = [394, 145, 28, 1]
class = 1
Shell weight <= 0.335
gini = 0.606
samples = 1935
value = [109, 987, 683, 156]
class = 2
Shell weight <= 0.067
gini = 0.339
samples = 418
value = [331, 76, 11, 0]
class = 1
Shell weight <= 0.056
gini = 0.599
samples = 150
value = [63, 69, 17, 1]
class = 2
Shell weight <= 0.168
gini = 0.551
samples = 1338
value = [106, 810, 364, 58]
class = 2
Shell weight <= 0.44
gini = 0.6
samples = 597
value = [3, 177, 319, 98]
class = 3
Length <= 0.278
gini = 0.151
samples = 196
value = [180, 15, 1, 0]
class = 1
Shucked weight <= 0.153
gini = 0.46
samples = 222
value = [151, 61, 10, 0]
class = 1
Height <= 0.102
gini = 0.574
samples = 112
value = [32, 64, 15, 1]
class = 2
Shucked weight <= 0.232
gini = 0.532
samples = 304
value = [73, 191, 38, 2]
class = 2
Shucked weight <= 0.406
gini = 0.538
samples = 1034
value = [33, 619, 326, 56]
class = 2
Shucked weight <= 0.446
gini = 0.583
samples = 389
value = [3, 148, 199, 39]
class = 3
Shucked weight <= 0.652
gini = 0.567
samples = 208
value = [0, 29, 120, 59]
class = 3
Shucked weight <= 0.058
gini = 0.235
samples = 119
value = [103, 15, 1, 0]
class = 1
Shell weight <= 0.082
gini = 0.485
samples = 141
value = [90, 46, 5, 0]
class = 1
Whole weight <= 0.377
gini = 0.395
samples = 81
value = [61, 15, 5, 0]
class = 1
gini = 0.0
samples = 77
value = [77, 0, 0, 0]
class = 1
gini = 0.346
samples = 36
value = [28, 8, 0, 0]
class = 1
gini = 0.176
samples = 83
value = [75, 7, 1, 0]
class = 1
gini = 0.39
samples = 75
value = [56, 17, 2, 0]
class = 1
gini = 0.539
samples = 66
value = [34, 29, 3, 0]
class = 1
gini = 0.219
samples = 41
value = [36, 4, 1, 0]
class = 1
gini = 0.524
samples = 40
value = [25, 11, 4, 0]
class = 1
gini = 0.314
samples = 38
value = [31, 5, 2, 0]
class = 1
gini = 0.588
samples = 59
value = [22, 30, 7, 0]
class = 2
gini = 0.53
samples = 53
value = [10, 34, 8, 1]
class = 2
Sex <= -0.5
gini = 0.534
samples = 186
value = [32, 118, 34, 2]
class = 2
Shell weight <= 0.146
gini = 0.495
samples = 118
value = [41, 73, 4, 0]
class = 2
Shell weight <= 0.248
gini = 0.587
samples = 620
value = [21, 326, 223, 50]
class = 2
Shell weight <= 0.26
gini = 0.436
samples = 414
value = [12, 293, 103, 6]
class = 2
Shell weight <= 0.364
gini = 0.571
samples = 78
value = [1, 10, 45, 22]
class = 3
Shell weight <= 0.404
gini = 0.555
samples = 311
value = [2, 138, 154, 17]
class = 3
Shell weight <= 0.507
gini = 0.551
samples = 84
value = [0, 5, 36, 43]
class = 4
Shell weight <= 0.558
gini = 0.487
samples = 124
value = [0, 24, 84, 16]
class = 3
Shell weight <= 0.135
gini = 0.502
samples = 93
value = [22, 61, 10, 0]
class = 2
Shell weight <= 0.136
gini = 0.546
samples = 93
value = [10, 57, 24, 2]
class = 2
Viscera weight <= 0.127
gini = 0.46
samples = 70
value = [18, 48, 4, 0]
class = 2
gini = 0.499
samples = 48
value = [23, 25, 0, 0]
class = 2
Shucked weight <= 0.244
gini = 0.516
samples = 416
value = [19, 260, 125, 12]
class = 2
Viscera weight <= 0.232
gini = 0.63
samples = 204
value = [2, 66, 98, 38]
class = 3
Length <= 0.562
gini = 0.272
samples = 104
value = [9, 88, 7, 0]
class = 2
Shucked weight <= 0.491
gini = 0.466
samples = 310
value = [3, 205, 96, 6]
class = 2
Shucked weight <= 0.557
gini = 0.534
samples = 225
value = [2, 111, 106, 6]
class = 2
Length <= 0.668
gini = 0.574
samples = 86
value = [0, 27, 48, 11]
class = 3
Diametre <= 0.553
gini = 0.406
samples = 89
value = [0, 18, 66, 5]
class = 3
gini = 0.521
samples = 43
value = [1, 6, 28, 8]
class = 3
gini = 0.591
samples = 35
value = [0, 4, 17, 14]
class = 3
gini = 0.579
samples = 49
value = [0, 5, 25, 19]
class = 3
gini = 0.431
samples = 35
value = [0, 0, 11, 24]
class = 4
gini = 0.607
samples = 35
value = [0, 6, 18, 11]
class = 3
(...)

Appendix E



IF Shell weight is less than or equal to 0.056 classify instance as class 1.

IF Shell weight is greater than 0.056 AND Height is greater than 0.102 classify instance as class 2.

IF Shell weight is greater than 0.056 AND Height is less than or equal to 0.102 classify instance as class 2.

Appendix F

Depth	9
Nodes	69
Leaves	35