ZZSC5836 Assessment 3, Option II

*Submitted in partial fulfilment of ZZSC5836

1st Paul John Cronin
Department of Science
University of NSW
Sydney, Australia
p.cronin@student.unsw.edu.au or 0000-0002-4531-1481

***Abstract*—This document considered the questions raised in Assessment 3, Option II: Tree and Ensemble Learning, of ZZSC5836 Machine Learning and Neural Networks. This assessment asked to classify abalone age by specific physical measurements, as well as the categorical variable of sex, into four different age classes. Decision trees with pre-pruning and post-pruning, ensemble methods such as bagging and pasting, as well as neural network models were employed to examine this question.**

***Index Terms*—decision tree visualisation, pre-pruning, post-pruning, bagging, random forests, neural networks.**

I. Introduction

Decision trees [1, Chapter 6] have been utilized for decades [2] for both regression and classification problems. This report looks to undertake a multi-class classification [1, Chapter 3] of the age of abalone into four classes using various decision trees [3] and ensemble methods to do so, and to compare the results with neural network [4] approaches. To achieve this abalone age classification, a number of physical measurements are supplied, those being linear measurements (abalone Length, Diameter and Height), mass measurement (abalone Whole, Shucked, Viscera and Shell mass) and the categorical classification of Sex (if the abalone is Male, Female or Infant).

The abalone age data is challenging to classify, because after age of 10 or 11 years, there is very little change in mass or shell measurements, making it difficult to differentiate between middle-aged abalone (11-15 years old) and geriatric abalone (greater than 15 years old).

Further, the data presented is class imbalanced, that is, there are far more younger abalone than older abalone. Therefore, it is important to consider this class imbalance to ensure equal classification accuracy of the different ages of abalone.

The author has studied this same dataset previously [5], but for linear regression estimation of the age, as opposed to the current problem of classification of the abalone age into four discrete age groups. It was found that the abalone age was linear for lineal measurements of the abalone (shell diameter, perimeter, etc), but non-linear for abalone mass measurements (shell mass, shucked mass, etc). Finally, from that study, there was found to be a great deal of variance for all measurements of abalone of the same age, no matter what the sex, size or maturity level of the abalone.

The motivation for this work was to explore various classifications techniques for a difficult dataset - can judicious choices of decision trees, neural networks and hyperparameters produce successful classification? What are the benefits and pitfalls of the various techniques?

Our major goals for this report are:

- Explore different libraries, classes and machine learning techniques to gain a deeper understanding of them, and
- Find under what conditions, and which optimized hyper-parameters, do the different techniques succeed or fail for this particularly difficult abalone dataset.

The paper is organised as follows: in section II is an exploration of the published work, in Section III the author details the Methodology used for the study, Section IV gives the results of the experimentation, and Section V is Discussion, in which the results are summarised and the implications of the results are interpreted.

II. Background and Related Work

A. Decision trees and visualisation

While there are many types of decision tree, such as CHAID [6] and Conditional Inference Trees [7], there are 4 basic decision trees algorithms:

- ID3 [8],
- C4.5 [9],
- C5.0 [10], and
- Classification And Regression Tree (CART) [11].

To get a sense of the differences between algorithms, Table is reproduced from Abedinia [12].

TABLE I
Comparison of decision trees

Features	**ID3**	**C4.5**	***CART***
Type of data	Categorical	Continuous and Categorical	Continuous and nominal attributes data
Speed	Low	Faster than ID3	Average
Boosting	Unsupported	Unsupported	Supported
Pruning	No	Pre-pruning	Post-pruning
Missing Values	Unsupported	Unsupported	Unsupported
Cost function	Info. entropy & info. gain	Info gain ratio	Gini impurity
Possible child leaves	2 or more	2	2

The most common functions used to split the data at each node of the decision tree are the Gini Impurity (1) and Entropy (3).

$$I_g(p) = 1 - \sum_{i=1}^{J} {p_i}^2 \tag{1}$$

$$H(T) = -\sum_{i=1}^{J} p_i \log_2 p_i \tag{2}$$

The CART algorithm, used in this paper, tries to minimize the following cost function [1, Chapter 6]:

$$J(k, t_k) = \frac{m_{\text{left}}}{m} I_{\text{left}} + \frac{m_{\text{right}}}{m} I_{\text{right}} \tag{3}$$

where $G_{\text{left/right}}$ are the measures of the Gini impurity of the left/right subset, and $m_{\text{left/right}}$ are the number of instances in the left/right subset.

Decision trees are somewhat unique amongst classifiers in that scaling and normalisation of the raw data is unnecessary [13].

B. Pre-pruning and post-pruning

While both pre-pruning and post-pruning decision trees remove under-powered decision nodes and leaves, they do so in such different manners as to be considered as separately.

Early-stopping [14] is a technique used by many models to prevent over-fitting, where a particular process of the model is stopped when the validation error reaches a minimum for that process. In the context of decision trees, early-stopping is known as 'pre-pruning'. Pre-pruning stops building out certain decision nodes early, while the decision tree is still being built.

One serious drawback with pre-pruning is the fact that it's a 'greedy algorithm'; once a node is stopped early the algorithm can't see beyond that node, possibly missing out on an advantageous partition.

Unlike pre-pruning, which stops building out different branches early, post-pruning first builds a complete, unconstrained decision tree, before removing leaves and nodes based upon a cost-complexity function[15][16], such as that found in (4).

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

Where T is the number of terminal nodes in the trees, and $R_{\text{alpha}}(T)$ is the misclassification rate of the terminal nodes. As the value for alpha increases, $R_{\text{alpha}}(T)$ changes, resulting in different optimal trees at that particular pruning. At a certain alpha, the overall error of the tree is minimized.

Post-pruning suffers from a two main issues:

- The full decision tree must first be built, which can be time-consuming, and
- For each snipped leaf, a sweep for alpha must be undertaken, and then repeated for the next leaf.

This results in a time consuming process, but which can result in better decision trees than found by pre-pruning.

C. Bagging of trees via random Forests

Bagging [17] stands for "Bootstrap AGGregatING" [18], another common technique in machine learning. In the context of decision trees, bagging is a way to obtain an ensemble of diverse classifiers [1, Chapter 7]. When samples are replaced, this process is called 'bagging' [19], but when samples are not replaced, it is called 'pasting' [20].

In the context of this paper, bagging and pasting is undertaken using the Scikit-Learn's BaggingClassifier [21].

D. Adam and SGD neural networks

Decision trees are quite a different machine learning approach to neural networks. In general, decision trees are extremely comprehensible, while neural networks can be quite obscure to a human [18]. Hence, these are quite independent classification processes, allowing for a fair comparison between them.

Neural Networks have a long and distinguished history [22]. Rosenblatt [23] invented the concept of the perceptron, a basic 'neuron' like structure that takes inputs, and processes those inputs in some manner, generating an output, which could be further fed into a network of other such neurons. The invention of 'back propagation', by Werbos [24] in 1974, pushed the field forward dramatically. Neural Networks are now a core element of machine learning.

The neural network used in this report is based around Scikit-Learn's MLPClassifier library [25]. This classifier allows for many hyperparameters to be tuned. These hyperparameters include the solvers, such as Stochastic Gradient Descent [26], Adaptive estimates of lower-order moments (ADAM) [27] as well as the activation functions relu [28],tanh, and logistic [29] functions.

In the context of this report, we shall use Scikit-Learn's GridSearchCV [30] to tune all these hyperparamters to determine the optimal solution, and then generate best neural network to be able to compare results to the decision tree and ensemble methods.

III. Methodology

A. General

In general, the tool-set used for this report was SciKit-Learn [31]. It provided the classes for Decision Trees, pre-pruning and post-pruning, bagging and pasting, and the neural network modeling.

Apart from published papers, the UNSW course notes and third-party websites were also valuable sources of information and guidance. These sites discuss and explain certain software packages in detail, such as Shaikh [32]. Where these websites offered guidance, they are referenced within this report, where appropriate.

The workflow of analysis for each model starts with a loading and cleaning the common dataset of abalone measurements, categories and ages. This data was split into a features dataset, and a target dataset. Each modelling process then split the features and target datasets into training and test subsets. Almost always, the training:test split was 60:40, except two

additional training sets were generated where training was 100% of the dataset (high variance) and another chosen from the best of 1,000 splits where the training was only 20%. These additional two datasets were to explore how the different decision tree models affected these extreme decision trees.

Once the training and test set were appropriately generated, the particular machine learning modality was performed, the results recorded and tabulated.

B. Pre-processing and EDA

The following continuous and categorical variables were provided in a dataset:

TABLE II
ABALONE MEASUREMENT VARIABLES

Variable	Type	Measure	*Description*
Sex	categorical	M/F/I	Sex description
Length	continuous	mm	Longest shell dim
Diameter	continuous	mm	perpendicular dim
Height	continuous	mm	with meat in shell
Whole weight	continuous	grams	whole abalone
Shucked weight	continuous	grams	weight of meat
Viscera weight	continuous	grams	gut weight
Shell weight	continuous	grams	after being dried
Age	integer	years	age in years

Prior to any data exploration or classification, it is important to check that the data is clean. The data was first pre-processed as followed:

- check for missing or NA data (none was found), and
- check for other possible errors, such as significant outliers (none was found).

This is an multi-class classification problem, where the abalone age falls into one of four classes: abalone 0-7 years of Age, 8-10 years of age, 11-15 years of age, and finally, abalone greater than 15 years of age. Before the different classification techniques were undertaken, the target variable Age was replaced with the AgeClass variable, reflecting these four distinct classes. Then, the single categorical variable Sex, which contained the three classes of Male, Female and Infant, was changed to a one-hot set of variables. The counts of each individual AgeClass was undertaken, and shown in Table III.

TABLE III
EXPLORATORY DATA ANALYSIS

Class	*1*	*2*	*3*	*4*
Age	0-7	8-10	11-15	>15
Count	839	1891	1186	261

One important aspect of this simple count analysis is the difference in count of the different classes - there are 1891 instances of Class 2 (8-10 years of age), compared to 261 instances of Class 4 (more than 15 years of age). Because of this greater than 7x difference in Count, it would be likely that the Class 4 classification accuracy would be lost. Hence it is vital to "balance the class data", either by:

1) reducing the instances of Classes with excess instances,
2) artificially increasing the under-represented Classes with copies of their instances, or
3) by weighting the Classes differently.

For this report, it was decided to go with 3). This was done usually by setting class_weights" to "balanced".

Further exploration of the data was undertaken with a simple Principal Component Analysis - see Figure 1. This is a common technique to understand the quality of the data. From Figure 1, it is clear the data is poor, with all the classes of abalone age overlapping. This would indicate poor accuracy performance with any classification tool, and one that would benefit from more sophisticated tools.

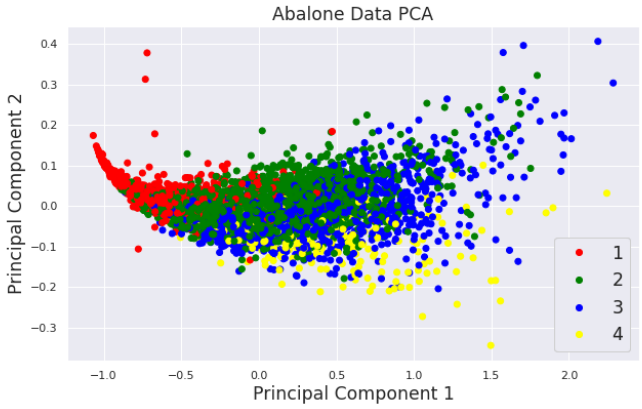

Fig. 1. Data exploration by Principal Component Analysis.

C. Decision trees and visualisation

The decision trees in this report will be based upon Scikit-Learn's implementation of the CART algorithm [33].

However, as this is a multi-class classification problem, where there are significantly different quantities of each class, we will always use "class balancing". This will give an over-all reduced classification accuracy, but a greater f1 score for each class.

The generated decision tree visualisation in this report was informed by the work of Johnston [34].

In general, all decision trees used the Scikit-Learn's DecisionTreeClassifier. Because of the fact the data classes were heavily imbalanced, the hyperparameter class_weight='balanced' was always used to compensate.

The assessment requested two elements for this section:

- Report the tree visualisation, and
- Translate a few selected nodes and leaves into If and Then rules.

To do so, it would be necessary to determine the hyperparameters for the decision tree. A Scikit-Learn GridSearchCV analysis was undertaken to find the decision tree hyperparameters:

- The initial depth of the decision tree (max_depth under SKLearn), and
- The minimum number of leaves in a sample (min_in_leaf under SKLearn).

This was attempted, but with poor results. The only free hyperparameter for the GridSearchCV was "cv", which refers to the cross validation splitting strategy. However, slightly different values of cv gave dramatically different hyperparameter results - see Table IV.

cv value	Max Depth	Min Samples Leaf
5	3	1
6	7	10
7	11	1
8	9	7
9	7	12
10	7	8
12	9	6
13	7	5
14	6	9
15	7	10

TABLE IV
INSTABILITY OF GRIDSEARCHCV - SLIGHT VARIATIONS OF CV CAUSE LARGE CHANGES IN HYPERPARAMETERS.

Clearly, from Table IV, the initial decision tree hyperparameter GridSearchCV were not sufficiently stable when determined by SKLearn's GridSearchCV.

In an attempt to gain greater insight, a custom hyperparameter grid search engine was coded, which allowed more flexibility with hyperparameters and many more experiments to average, achieving more stability.

Under this custom grid search, the hyperparameter for tree depth was varied between 1 and 20, while the hyperparameter for minimum leaf samples was varied from 1 to 15. For each set of the hyperparameters, 100 different experiments were run, where every different hyperparameter set saw the same 100 different train/test splits. For each hyperparameter set the mean accuracy was calculated and the results given in Figure 2.

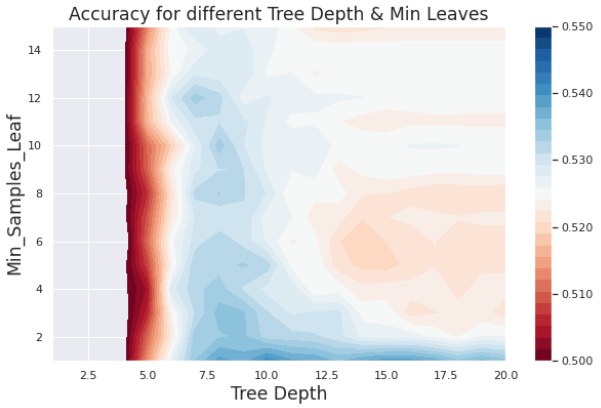

Fig. 2. The relatively flat plateau of accuracy causes instability in GridSearchCV.

As can be seen in Figure 2, for a decision tree with a depth of 4 or less, the classification accuracy falls away quickly. This is due to the fact that Class 4 abalone won't be accurately classified as they are so dramatically under-represented.

Above a decision tree depth of 6, there is a "plateau" of equal classification accuracy, with very slight regional fluctuations. Recall that this is with 100 experiments undertaken for each combination of hyperparameters. This explains why the SKLearn GridSearchCV algorithm, without the moderating influence of 100 experiments, tended to fail - any slight error causes wildly different hyperparameters to be chosen. The standard GridSearchCV approach was too unstable to find the maximum probability location.

There seems to be two relatively higher accuracy in Figure 2 with slightly higher probability. They are:

- Around a decision tree depth of 8, independent of the min_sample_leaf, and
- when min_samples_leaf is 1, and the decision tree depth exceeds 7.

For the region described by min_samples_leaf equal to 1, it has a slightly higher probability as it is capturing the Class 4 abalone. However, that hyperparameter was rejected because it was felt that it was over-fitting the model. Hence, the final hyperparameters chosen for the initial decision tree are for max_tree_depth = 8 and min_sample_leaf = 2.

As a final validation that this was an acceptable set of hyperparameters, a set of 100 experiments were undertaken, with min_sample_leaf = 2 and max_tree_depth varying from 1 to 20. A comparison of overall mean accuracy vs decision tree depth is given in Figure 3.

Fig. 3. Training and test accuracy for various decision tree depths.

As expected, a relatively good classification accuracy of 0.534 occurs at a max_tree_depth = 8. Additionally, the accuracy of measurement for each AgeClass is broken out in Figure 2.

Interpreting Figure 4 we have:

- A high initial accuracy for the AgeClasses of 1 and 3, for a single binary split - all the data must fall into one of two buckets. Note how AgeClass 3 falls away quickly until it level off at approximately Age = 17.
- At a tree depth of 3, AgeClass 2 is first resolved, allowing for a non-zero probability of correct classification.
- At the tree depth of 4, AgeClass 4 is first resolved, allowing for a non-zero probability of correct classification for that class. The accuracy of correct classification of

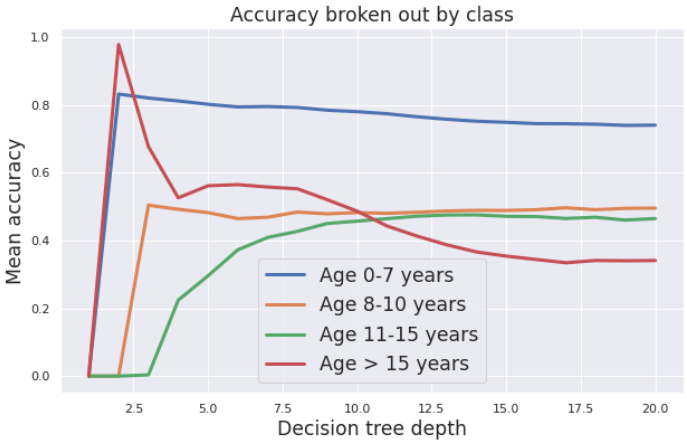

Fig. 4. Accuracy of test data for decision tree, broken out by class.

this class will continue to increase till approximately Age = 11.

Considering the mean accuracy of all the Classes, an initial decision tree depth of 8 is an acceptable compromise for all the different Classes, and is what was used for the computation of the decision tree visualisation and IF / THEN rules. This was designated a 'standard' variance decision tree.

While not mandated by the assessment, two additional decision trees were built, to understand how the various decision tree methods were affected by the bias-variance trade-off. These additional trees were constructed with the following parameters to generate what was designated 'high' and 'low' variance decision trees:

- a decision tree formed when trained with 100% of the data. Obviously, this tree would be maximally over fitted and suffer from a strong variance.
- by choosing the best decision tree out of 1,000, formed by training with only 20% of the training set. This would be considered to have a low variance decision tree.

D. Pruning decision trees

In general, pruning decision trees [1] is a set of techniques to reduce the variance of decisions trees. They fall into two broad categories:

- pre-pruning, also known as early stopping, where certain unnecessary nodes are not added during the decision tree building process.
- post-pruning, where the decision tree is first fully built, and then unnecessary nodes removed, and

Pre-pruning is often considered more efficient, but not necessarily as mathematically rigorous, as it does not require the full decision tree to be built before optimization [35].

E. Pre-pruning

Pre-pruning involves generating an unconstrained decision tree, and then performing "early-stopping" when a certain criteria is met [35][36][37]. Krueger [37] uses two different early stopping criteria, those being:

- min_samples_split
- min_samples_leaf

For this report, both of these criteria were used, designated as 'split' and 'leaf'. For each of these criteria, a DecisionTreeClassifier was constructed, which was class weight balanced, and having a Gini impurity mechanism. The following hyperparameters were allowed to vary:

- min_samples_split varied from 0.025 to 1, with steps of 0.025, and
- min_samples_leaf varied from 0.025 to 0.5 with steps of 0.025.

For each possible outcome the accuracy_score was computed, and the best chosen. For that best chosen decision tree, it's various accuracy scores (accuracy, weighted precision, weighted recall and weighted f1-score) were computed and recorded. The decision tree was also computed, displayed and saved.

This allowed for comparison of 'split' and 'leaf' with the performance of the 'standard', 'high variance' and 'low variance' decision trees.

F. Post-pruning

Post-pruning of the 'standard', 'high variance' and 'low variance' decision trees were undertaken, based around the Sci-Kit Learn's plot_cost_complexity_pruning [38] class, and following examples such as given by Singh [36].

Post-pruning first generates the full decision tree, and then prune back poor quality leaves to minimize the cost-complexity function. This cost complexity function has the hyperparameter alpha, and we have the goal of finding the alpha that maximises the overall accuracy.

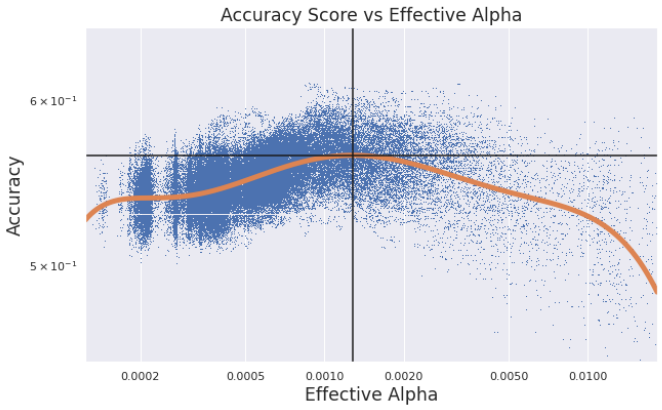

Fig. 5. The result of 200 post-pruning decision tree experiments to determine the optimal effective alpha.

Figure 5 shows the results of that search for the optimal alpha. It is generated from 200 such experiments, where each alpha was computed for each experiment, and then the classification accuracy was computed for each decision tree corresponding

to each alpha for each of the 200 experiments. Each of these results is plotted as a blue dot.

As can be seen from the Figure, it is a very noisy result, where alpha spans orders of magnitude. A polynomial fit of order 7 was fit to the alpha / classification accuracy data, to find the best effective alpha = 0.00128 with an expected classification accuracy of 0.562, but with a knowledge that the accuracy could have a large standard deviation of approximately 0.015. The decision tree resulting from this search for an optimal effective alpha is given in the Results.

G. Bagging of trees via random Forests

The process of bagging via random forests utilised Scikit-Learn's BaggingClassifier and DecisionTreeClassifier. First, a DecisionTreeClassifier was build, with class_weight "balanced". A BaggingClassifier was built around the DecisionTreeClassifier, where the following hyperparameters were allowed to vary:

- the number of estimators from 2, 4, 8, 16, 32, 64, 128, 256, 512 to 1024.
- the number of samples from 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 to 2048.

The optimal number of samples was chosen to be 200, which was subsequently used to find how the mean classification accuracy and standard deviation improved as the number of estimators increased.

H. Adam and SGD neural networks

Following the assessment guideline, a SKLearn GridSearchCV was undertaken using an MLP neural network classifier [39], in order to compare to the Decision Tree and Random Forest results. The following hyperparameter ranges were searched:

- Solver: SGD and Adam,
- Activation function: logistic, tanh, and relu.
- The number of hidden dimensions being 1, 2 and 3.
- Hidden dimension sets of (20), (40), (20,20), (10,20,10), and (10,30,10).
- The learning rate being constant or adaptive.
- The value for alpha being 0.001,0.01, 0.1.

Additionally, the value for cv was set as 5 and accuracy_score was used as the scorer.

In total, 540 different combinations of hyperparameters were attempted, and the results for each were recorded.

IV. Results

In this section, we present the results of the different modelling techniques: the standard decision tree visualisation, both pre-pruning and post-pruning of the decision trees, bagging and pasting, as well as the results of the comparison with neural networks.

A. Decision trees and visualisation

After determining the optimal hyperparameters from custom grid search code, those being a decision tree depth of 8 and min_samples_leaf of 2, a 'standard' decision tree was found - see the visualisation in Figure 6. This tree is 8 deep, but only the first 5 levels are displayed for clarity [40]. It is clear from this figure, a single class is split fairly cleanly to the left, and the remaining three classes are much more intermixed. Apart from one last node, every decision node has two leaves. For clarity, the 'high' and 'low' variance decisions trees are not shown, but are available in the accompanying Jupyter Notebook. Further, the first four levels of If/Then rules were generated for the 'standard' decision tree - see Figure 7. Finally, the confusion matrix was generated for the 'standard' decision tree - see Figure 10. Clearly, and not unexpectedly, the worst performing class was the geriatric abalone, those over 15 years of age.

Table V gives the statistics for the 'standard', 'high variance' and 'low variance' decision trees, all with depth of 8 and minimum split of 2. The 'standard' variance decision tree, which used 60% of the data for the training, will be used as the comparative decision tree for the remaining analysis. 10 different experiments were run for this decision tree, and the mean and standard deviation for the different statistics are given.

As there is only one 'high variance' and 'low variance' decision tree possible, the standard deviation for each statistic could not be computed.

Statistic	Std Var	High Var	Low Var
No Nodes	279	343	219
Max Depth	7	8	8
Accuracy	0.548 / 0.015	0.660	0.587
Precision	0.575 / 0.013	0.683	0.595
Recall	0.548 / 0.015	0.660	0.587
f1-score	0.552 / 0.015	0.661	0.590
>15 years	0.290	0.410	0.274

TABLE V
STATISTICS FOR THE 'STANDARD VARIANCE', THE 'HIGH VARIANCE' DECISION TREE AND THE 'LOW VARIANCE' DECISION TREES.

B. Pre-pruning the decision tree

This section demonstrates and measures the performance improvement from pre-pruning the three decision trees described in Section IV-A.

As such, the following tables report the results for 'split' and 'leaf' early stopping for the original 'standard variance' decision tree, and the pre-pruned 'split' and 'leaf' decision trees.

Examining VI it is clear that the 'standard variance' decision tree, the pre-pruned 'split' decision tree and pre-pruned 'leaf' decision tree are approximate, with the pre-pruned 'split' decision tree performing marginally better.

Where the benefit of pre-pruning decision trees exceeds normal decision tree construction is in the number of nodes in the tree. Early stopping for the 'split' decision tree matches

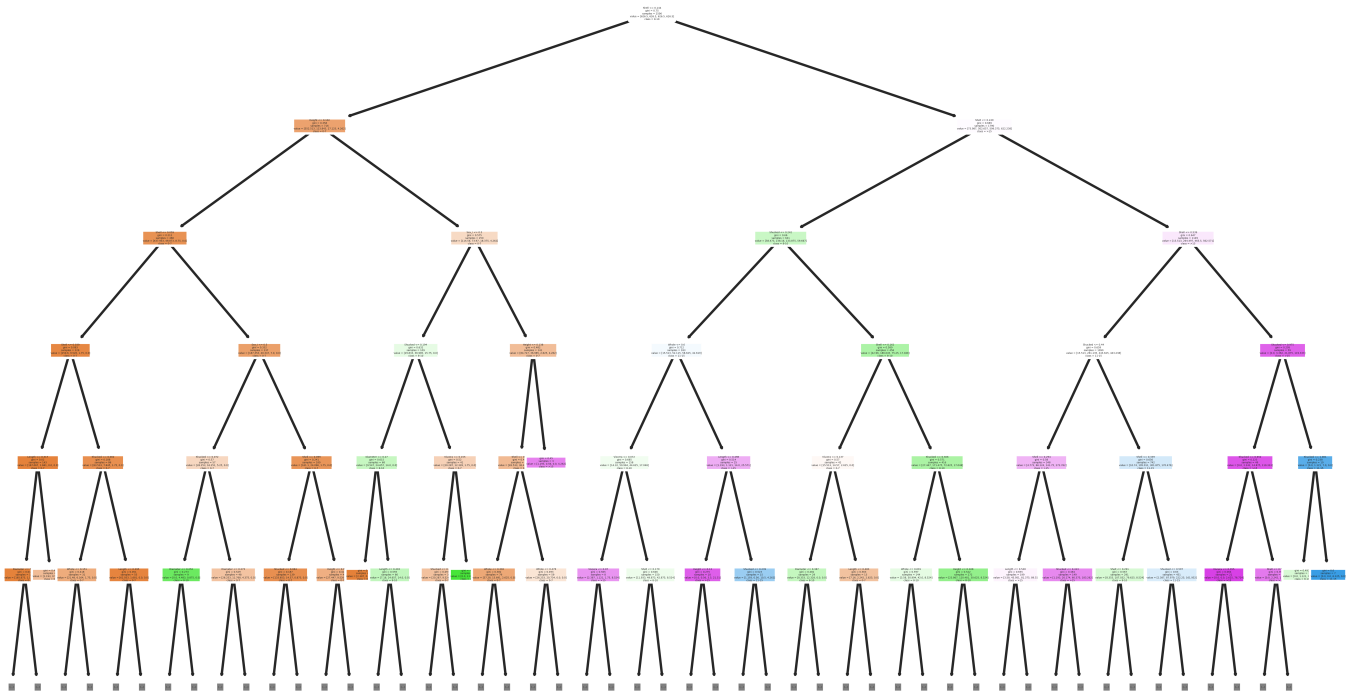

Fig. 6. Visualisation for the 'standard' decision tree, showing only the first 5 of 8 levels.

```
|--- Shell <= 0.14
|   |--- Height <= 0.10
|   |   |--- Shell <= 0.06
|   |   |   |--- Shell <= 0.04
|   |   |   |   |--- truncated branch of depth 5
|   |   |   |--- Shell >  0.04
|   |   |   |   |--- truncated branch of depth 5
|   |   |--- Shell >  0.06
|   |   |   |--- Sex_I <= 0.50
|   |   |   |   |--- truncated branch of depth 5
|   |   |   |--- Sex_I >  0.50
|   |   |   |   |--- truncated branch of depth 5
|   |--- Height >  0.10
|   |   |--- Sex_I <= 0.50
|   |   |   |--- Shucked <= 0.19
|   |   |   |   |--- truncated branch of depth 5
|   |   |   |--- Shucked >  0.19
|   |   |   |   |--- truncated branch of depth 5
|   |   |--- Sex_I >  0.50
|   |   |   |--- Height <= 0.14
|   |   |   |   |--- truncated branch of depth 5
|   |   |   |--- Height >  0.14
|   |   |   |   |--- weights: [1.19, 0.56, 0.00, 4.26] class: 4
|--- Shell >  0.14
|   |--- Shell <= 0.25
|   |   |--- Shucked <= 0.24
|   |   |   |--- Whole <= 0.60
|   |   |   |   |--- truncated branch of depth 5
|   |   |   |--- Whole >  0.60
|   |   |   |   |--- truncated branch of depth 5
|   |   |--- Shucked >  0.24
|   |   |   |--- Shell <= 0.16
|   |   |   |   |--- truncated branch of depth 5
|   |   |   |--- Shell >  0.16
|   |   |   |   |--- truncated branch of depth 5
|   |--- Shell >  0.25
|   |   |--- Shell <= 0.54
|   |   |   |--- Shucked <= 0.44
|   |   |   |   |--- truncated branch of depth 5
|   |   |   |--- Shucked >  0.44
|   |   |   |   |--- truncated branch of depth 5
|   |   |--- Shell >  0.54
|   |   |   |--- Shucked <= 0.98
|   |   |   |   |--- truncated branch of depth 5
|   |   |   |--- Shucked >  0.98
|   |   |   |   |--- truncated branch of depth 2
```

Fig. 7. If / then rules for three levels of the standard decision tree.

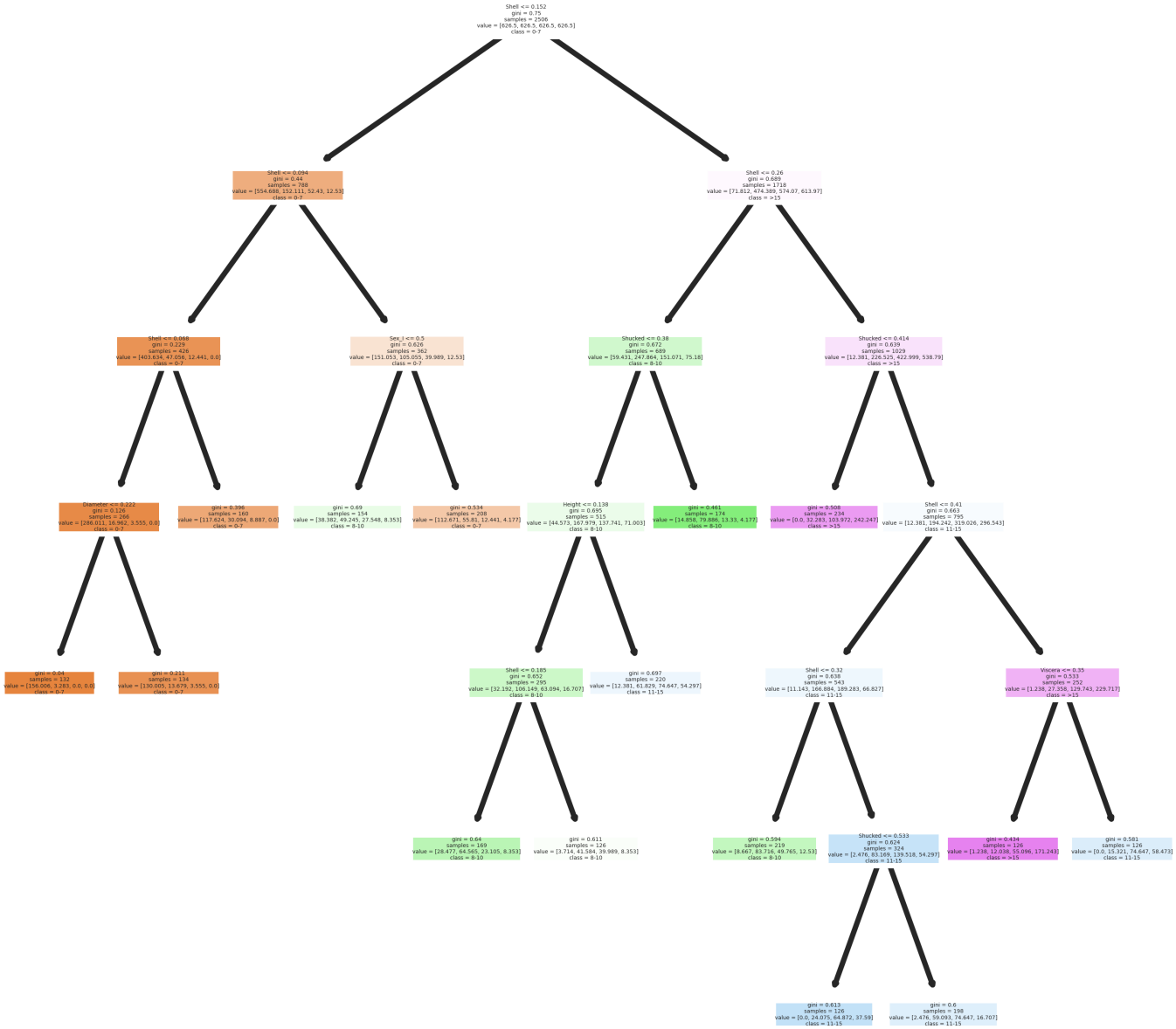

Fig. 8. Visualisation for the entire 'standard' decision tree with pre-pruning, with early stopping due to min_samples_leaf criteria.

Statistic	Std Var	Split	Leaf
No Nodes	279	47	29
Max Depth	7	8	6
Accuracy	0.548	0.562	0.559
Precision	0.575	0.605	0.585
Recall	0.548	0.562	0.559
f1-score	0.552	0.571	0.562
>15 years	0.290	0.247	0.272

TABLE VI
STATISTICS FOR THE 'STANDARD VARIANCE', THE 'SPLIT' DECISION TREE AND THE 'LEAF' PRE-PRUNED DECISION TREES.

the 'standard' performance, but does so using 47/279 = 16.8% of the nodes. Even better, the early stopping by 'leaf' criteria gives superior performance with only 29/279 = 10.4% of the nodes as the 'standard' decision tree.

The full pre-pruned 'leaf' decision tree is reproduced in Figure 8.

Hence, the careful choice of early-stopping parameter can train a complex decision tree faster (as it doesn't have to fully build) and give superior results.

It should be noted, for various experiments, sometimes the 'leaf' result was superior to the 'split' result, but at other times, the results were reversed. Hence, it is important to always do both.

For brevity, corresponding tables for the 'high variance' and 'low variance' are available in the accompanying Jupyter Notebook, but not reproduced in this report. But to summarize their results, the performance of the 'high variance' decision

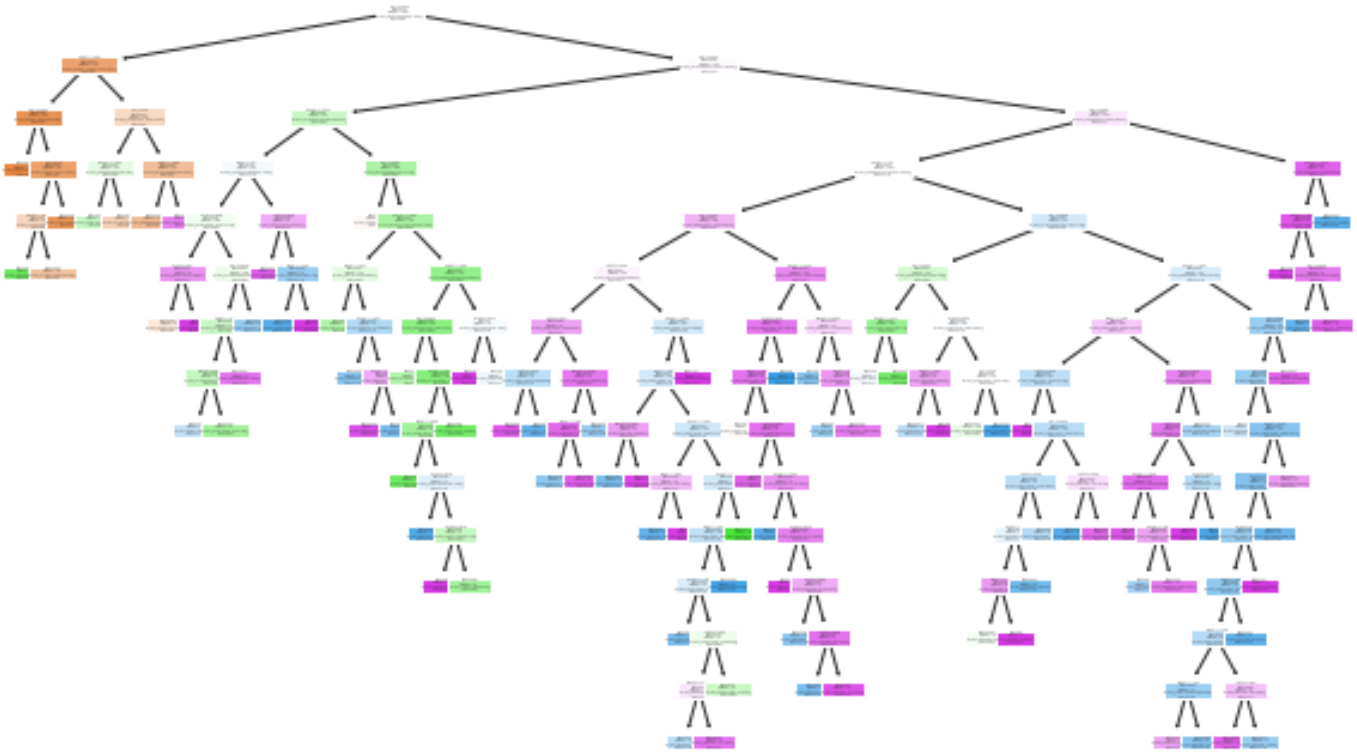

Fig. 9. Visualisation for entire post-pruned decision tree.

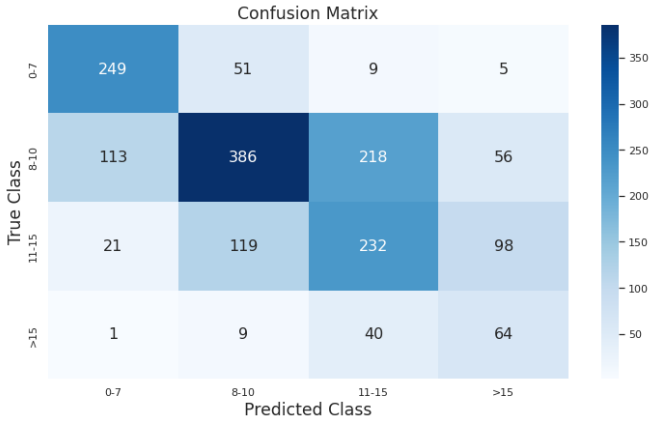

Fig. 10. Confusion matrix for the 'standard' decision tree.

tree suffers significantly (accuracy score falling from 0.66 to 0.579 and 0.512 for 'split' and 'leaf' respectively), while the 'low variance' decision tree's performance was approximately the same.

C. Post-pruning the decision tree

Following the determination of an optimal effective alpha, and predicting a range of possible inaccuracy given that alpha, 10 experiments with that effective alpha were undertaken, giving the mean accuracy as 0.557 with a standard deviation of 0.016, very similar to those predicted in the search for the optimal alpha III-F.

One of those post-pruned decision trees is reproduced in full in Figure 9.

Table VII compares the 'standard' variance decision tree with a post-pruned decision tree, generated with the optimal effective alpha. The post-pruned decision tree gives slightly better scores, but using only 179/279 = 64.2% of the nodes.

Curiously, the post-pruned decision tree does have 2x the depth of the 'standard variance' decision tree. The use of the cost-complexity pruning does not force some branches to go the full depth, but does allow other branches to go far deeper - something the standard decision tree does not allow.

Comparing the post-pruned decision trees to the pre-pruned decision trees in Table VI, the post-pruning decision tree is found to be very similar, and so it could well be argued that the extra effort in building out the full decision tree, as well as searching for the optimal effective alpha, before snipping back leaves, is not worth it in comparison to the far simpler pre-pruning approach.

D. Bagging and pasting of trees via random Forests

Bagging refers to "bootstrap aggregation" with replacement, while pasting refers to "bootstrap aggregation" without replacement. In this section, the author reports on the outcome of bagging and pasting trees of the abalone dataset via random forests.

Statistic	Std Var	Postpruned
No Nodes	279	179
Max Depth	7	14
Accuracy	0.548 / 0.015	0.556
Precision	0.575 / 0.013	0.585
Recall	0.548 / 0.015	0.556
f1-score	0.552 / 0.015	0.557
>15 years	0.290	0.239

TABLE VII
COMPARATIVE METRIC FOR THE 'STANDARD VARIANCE' DECISION TREES AND THE POST-PRUNED DECISION TREE.

For this report, the Sci-Kit Learn BaggingClassifier was used, with the number of estimators tested being varying as follows: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 and 2048.

While the assessment calls for 10 experiments to be undertaken for each number of estimators, to improve the statistics, the number of experiments was increased to 100.

Figure 11 shows the effect on classification accuracy as the number of estimators increases on both bagging and pasting; it improves from 0.545 at 2 estimators, and approaches a plateau maximum of of 0.637 around 512 estimators. Clearly, both bagging and pasting give identical results for this dataset. Any additional estimators shows no significant improvement in accuracy for either bagging or pasting.

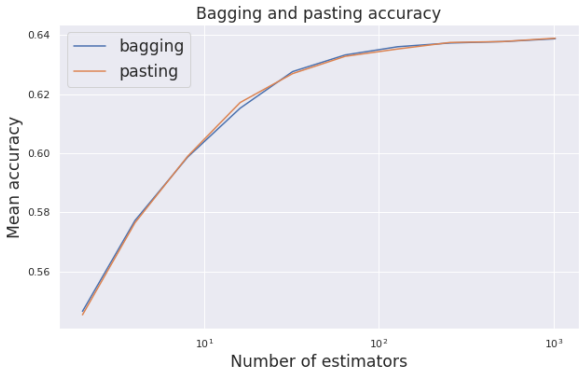

Fig. 11. Bagging and pasting classification accuracy.

As with the classification accuracy estimation, Figure 12 the standard deviation of bagging and pasting is identical as the number of estimators increase. The standard deviation of both decreases from a maximum of 0.018, when there were 2 estimators and finds a minimum standard deviation of 0.01 when the number of estimators reaches 128. Further increasing the number of estimators above this value does little to improve the standard deviation.

It should be noted that a long experiment was undertaken, with the number of experiments per estimator was increased from 100 to 2000. Under those conditions, the bagging and pasting accuracy were still identical, as was the standard deviation as a function of estimators.

The conclusion of this experiment is that it does not seem to matter for this dataset whether bagging or pasting is used, as they appear to give identical results.

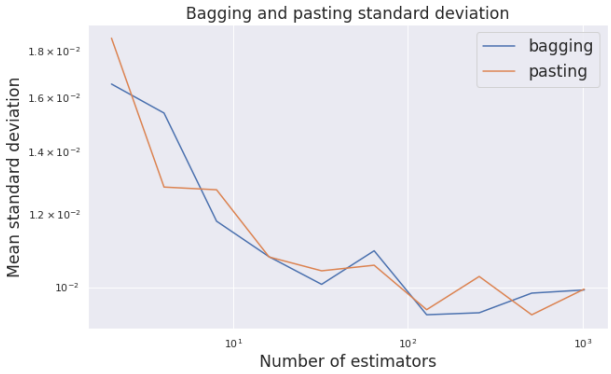

Fig. 12. Bagging and pasting classification standard deviation.

If we comparison the standard decision tree to the bagging and pasting ensembles classifiers, we find that both bagging and pasting give significantly better results. For data comprising 100 experiments, the statistics show that pasting gives a very slightly improved result over bagging, but this improvement is well within a single standard deviation.

Statistic	Std Var	Bagging	Pasting
Accuracy	0.548 / 0.015	0.631 / 0.011	0.632 / 0.011
Precision	0.575 / 0.013	0.630 / 0.012	0.634 / 0.014
Recall	0.548 / 0.015	0.631 / 0.011	0.632 / 0.011
f1-score	0.552 / 0.015	0.610 / 0.013	0.612 / 0.013
>15 years	0.290	0.800	0.667

TABLE VIII
STATISTICS FOR THE 'STANDARD VARIANCE' DECISION TREE AS WELL AS FOR BAGGED AND PASTED ENSEMBLE APPROACHES.

E. Adam and SGD neural networks

An MLP neural network was developed with a number of different hyperparameters. The solution space was searched by GridSearchCV with the following results for the best neural network:

- Mean accuracy: 0.648 with standard deviation of 0.0075,
- Solver being Adam,
- Activation function being relu,
- Hidden layers being three, with the number of nodes at each hidden layer (10,20,10),
- an alpha of 0.01.

If we consider the top 15 neural networks, out of the 540 results explored, it is found that:

- Adam was chosen 13/15 times for best solver, while SGD was only chosen 2/15,
- The tanh activation function was chosen 8/15 times, while relu was chosen 7/15. A logistic activation function was never selected in the top 15 results.
- The most net with three hidden layers (10,20,10) was chosen 8/15 times, while two hidden layers (20,20) was chosen 7/15 times. The single hidden layer (40) was not chosen at all,

- a constant learning rate was chosen 9/15 times, while an adaptive learning was only chosen 6/15 times, and finally,
- an alpha of 0.001 was chosen 8/15 times, while an alpha of 0.01 was chosen 7/15 times.

A set of 10 experiments were undertaken with the optimal neural network, giving the following statistics:

Statistic	Std Var	Neural Net
Accuracy	0.548 / 0.015	0.644 / 0.010
Precision	0.575 / 0.013	0.640 / 0.009
Recall	0.548 / 0.015	0.644 / 0.010
f1-score	0.552 / 0.015	0.631 / 0.012
>15 years	0.290	0.450

TABLE IX
STATISTICS FOR THE 'STANDARD VARIANCE' DECISION TREE AS WELL FOR A NEURAL NETWORK APPROACH.

V. DISCUSSION

A. Decision trees and visualisation

The 'standard' decision tree was the baseline for comparison for all other models. By examining Table V it is found, not unsurprisingly, that the 'high variance' decision tree, which was trained with 100% of the data had the highest accuracy of all decision trees, while the 'low variance' decision tree, chosen from 1,000 experiments with only 20% of the data used as training, performed somewhat less, but not dramatically so. Included in the data is the precision score for Class 4 - the geriatric abalone over 15 years of age. It will have the poorest precision of all the classes, and is a good proxy for how well the model did overall. The standard decision tree only classifies it accurately 29% of the time, not much better than random. The over-fitted high-variance decision tree did somewhat better, and surprisingly, the low variance decision tree did poorly.

This poor classification accuracy of Class 4 could be either a limit of the methodology, perhaps more weighting could have given better results, or more likely it is a limitation of the data itself.

B. Pre-pruning

For pre-pruning, two different stopping criteria were examined, those being 'split' and 'leaf'. Comparing the results of Table VI each of these pre-pruning stopping criteria to the standard decision tree, it is found to be a significant improvement in nearly all ways - the complexity of the decision trees are dramatically reduced, with a slight increase in accuracy, precision, recall and f1-score overall. However, the classification precision for abalone over 15 years of age declined.

Hence, pre-pruning seems to be an improvement over a simple decision tree - it is faster, simpler and gives improved results, but in cases of poor-quality data, or where the data is heavily unbalanced, it can reduce the classification accuracy of the under-represented data.

This pre-pruning step was also undertaken for the high and low variance decision trees - see the accompanying Jupyter Notebook. The high variance decision tree metrics all dropped when pre-pruned, but the low variance decision tree metrics stayed approximately equal. The high variance change is to be expected - any change to its brittle structure will cause dramatic classification accuracy reduction.

C. Post-pruning

As can be seen from Table VII, the improvement due to post-pruning is no better than pre-pruning, and in some areas, such as the classification accuracy of Class 4 abalone, significant worse. This is not to say that post-pruning is always worse than pre-pruning, it could just be this particular dataset.

D. Bagging & pasting of trees via random Forests

While there was found little difference between the results of Bagging and Pasting, see Table VIII, there was significant improvement of both methods over the standard decision tree, the best of all methods. Even more curiously, the classification accuracy of Class 4 quite dramatically improved, from sum 0.300 to 0.667 for pasting and 0.800 for bagging. In fact, for both bagging and pasting, Class 4 has become either the most, or second most accurately classified classes. Hence, as opposed to pre-pruning and post-pruning, which did not handle multi-class unbalanced datasets well, both bagging and pasting excelled.

It is unclear why bagging and pasting gave almost identical mean and standard deviations as the number of estimators increased, perhaps it was because of the nature of the data or possibly that the number of samples drawn were small compared to the overall dataset size.

E. Adam and SGD neural networks

The final comparison is to an unrelated modelling method - a simple MLL neural network, see Table IX. Such a network trains in a time comparable to post-pruning, but it exceeds all of the decision tree and ensemble methods. Only bagging and pasting come close, and actually exceed the neural network in accuracy for Class 4 classification accuracy.

This neural network used only 40 hidden nodes, best distributed in three hidden layers (10,20,10). It is certainly possible that a slightly different distribution of nodes

VI. CONCLUSIONS

This report was to compare and contrast various machine-learning methods, such as decision trees, ensemble methods and neural networks, to classify the age of abalone from physical measurements of the abalone, and knowledge of the abalone sex.

The abalone data is of extremely poor quality, with the different classes intermingled intimately within the data-space, as well as being significantly unbalanced (the least populated class was 1/7 of the most populace class).

Under these circumstance, it was found that decision trees alone were only moderately successful in accurately classifying the data, with the least abundant class of abalone being extremely poorly classified.

Techniques such as pre-pruning and post-pruning improved the accuracy marginally, but the extra time cost associated with

Statistic	Std Var	High Var	Low Var	Preprune Split	Preprune Leaf	Postprune	Bagging	Pasting	NN
No Nodes	279	343	219	47	29	179	-	-	40
Max Depth	7	8	8	8	6	14	-	-	3
Accuracy	0.548 / 0.015	0.660	0.587	0.562	0.559	0.556	0.631 / 0.011	0.632 / 0.011	0.644 / 0.010
Precision	0.575 / 0.013	0.683	0.595	0.605	0.585	0.585	0.630 / 0.012	0.634 / 0.014	0.640 / 0.009
Recall	0.548 / 0.015	0.660	0.587	0.562	0.559	0.556	0.631 / 0.011	0.632/0.011	0.644 / 0.010
f1-score	0.552 / 0.015	0.661	0.590	0.571	0.562	0.557	0.610 / 0.013	0.612/0.013	0.631 / 0.012
>15 years	0.290	0.410	0.274	0.247	0.272	0.239	0.800	0.667	0.450

TABLE X
STATISTICS FOR ALL RESULTS FOR SIDE-BY-SIDE COMPARISON OF APPROACHES.

post-pruning was not worth it. The ensemble techniques of bagging and pasting produced a significant increase in classification accuracy, increasing classification accuracy approximately 5-8% for different accuracy metrics, except for the least popular class, which was vastly improved by more than 30 or 40%.

However, a relatively simple neural network, of only 40 nodes, still outperformed all the decision trees and associated ensemble methods.

Future research directions would be to explore different forms of Boosting, such as AdaBoost, GradientBoost, and XGBoost, which have the potential to significantly improve classification accuracy.

ACKNOWLEDGMENT

The author would like to acknowledge Dr Rohitash Chandra, the course instructor for ZZSC5836 at the University of New South Wales. The Jupyter Notebook that accompanies this report was significantly assisted by the course work [18].

REFERENCES

[1] A. Geron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, English, 2 edition. Beijing China ; Sebastopol, CA: O'Reilly Media, Inc, USA, Oct. 2019, ISBN: 9781492032649.

[2] W.-Y. Loh, "Fifty years of classification and regression trees," *International Statistical Review*, vol. 82, pp. 329–348, 3 Dec. 2014, ISSN: 03067734. DOI: 10.1111/insr.12016.

[3] R. Messenger and L. Mandell, "A modal search technique for predictive nominal scale multivariate analysis," *Journal of the American statistical association*, vol. 67, no. 340, pp. 768–772, 1972.

[4] C. Clabaugh and J. Pang, *Neural Networks - History*, 2000. [Online]. Available: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/index.html (visited on 06/15/2022).

[5] P. Cronin, "Zzsc5836 assessment 2, option i," University of New South Wales, May 2022.

[6] G. V. Kass, "An exploratory technique for investigating large quantities of categorical data," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 29, no. 2, pp. 119–127, 1980.

[7] T. Hothorn, K. Hornik, and A. Zeileis, "Unbiased recursive partitioning: A conditional inference framework," *Journal of Computational and Graphical statistics*, vol. 15, no. 3, pp. 651–674, 2006.

[8] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1 Mar. 1986, ISSN: 0885-6125. DOI: 10.1007/BF00116251.

[9] J. R. Quinlan, "Improved use of continuous attributes in c4. 5," *Journal of artificial intelligence research*, vol. 4, pp. 77–90, 1996.

[10] M. Kuhn, K. Johnson, *et al.*, *Applied predictive modeling*. Springer, 2013, vol. 26.

[11] L. Breiman, J. Friedman, R. Olshen, and C. Stone, "Classification and regression trees. wadsworth & brooks," *Cole Statistics/Probability Series*, 1984.

[12] A. Abedinia, *Survey of the Decision Trees Algorithms (CART, C4.5, ID3)*, en, Sep. 2020. [Online]. Available: https://medium.com/@abedinia.aydin/survey-of-the-decision-trees-algorithms-cart-c4-5-id3-97df842831cd (visited on 06/18/2022).

[13] Nik, *Decision Tree Classifier with Sklearn in Python • datagy*, en-US, Apr. 2022. [Online]. Available: https://datagy.io/sklearn-decision-tree-classifier/ (visited on 06/16/2022).

[14] O. N. Strand, "Theory and methods related to the singular-function expansion and landweber's iteration for integral equations of the first kind," *SIAM Journal on Numerical Analysis*, vol. 11, no. 4, pp. 798–825, 1974.

[15] J. R. Quinlan, "Simplifying decision trees," *International journal of man-machine studies*, vol. 27, no. 3, pp. 221–234, 1987.

[16] S. Arora, *Cost Complexity Pruning in Decision Trees — Decision Tree*, en, Oct. 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/10/cost-complexity-pruning-decision-trees/ (visited on 06/17/2022).

[17] M. (https://stats.stackexchange.com/users/268106/msallal), *When should the pasting ensemble method be used instead of bagging?* Cross Validated, URL:https://stats.stackexchange.com/q/440039 (version: 2020-06-11). eprint: https://stats.stackexchange.com/q/440039. [Online]. Available: https://stats.stackexchange.com/q/440039.

[18] R. Chandra, *ZZSC5836 (H3 2022) – Ed Lessons*, 2022. [Online]. Available: https://edstem.org/au/courses/8454/lessons (visited on 06/17/2022).

[19] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[20] ——, "Pasting small votes for classification in large databases and on-line," *Machine learning*, vol. 36, no. 1, pp. 85–103, 1999.

[21] Scikit-Learn, *Sklearn.ensemble.BaggingClassifier*, en, 2022. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html (visited on 06/19/2022).

[22] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[23] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[24] P. Werbos, "Beyond regression:" new tools for prediction and analysis in the behavioral sciences," *Ph. D. dissertation, Harvard University*, 1974.

[25] Scikit-Learn, *Sklearn.neural_network.MLPClassifier*, en, 2022. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.neural_network.MLPClassifier.html (visited on 06/19/2022).

[26] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[28] K. Fukushima, "Visual feature extraction by a multilayered network of analog threshold elements," *IEEE Transactions on Systems Science and Cybernetics*, vol. 5, no. 4, pp. 322–333, 1969.

[29] G. Hinton, L. Deng, D. Yu, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[30] Scikit-learn, *Sklearn.model_selection.GridSearchCV*, en, 2022. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.model_selection.GridSearchCV.html (visited on 06/19/2022).

[31] scikit-learn. "Scikit-learn: Machine learning in python — scikit-learn 1.1.1 documentation." (), [Online]. Available: https://scikit-learn.org/stable/ (visited on 06/16/2022).

[32] T. Shaikh, *Useful scikit-learn Methods*, en, 2015. [Online]. Available: https://towardsdatascience.com/random-useful-scikit-learn-methods-a032f78e1ec3 (visited on 06/16/2022).

[33] scikit-learn, *1.10. Decision Trees*, en, 2022. [Online]. Available: https://scikit-learn/stable/modules/tree.html (visited on 06/16/2022).

[34] B. Johnston, *Supervised-Learning-with-Python*, original-date: 2018-11-06T13:23:46Z, Apr. 2019. [Online]. Available: https://github.com/TrainingByPackt/Applied-Supervised-Learning-with-Python/blob/f125cecde1af4f77017302c3393acf9c2415ce9a/Chapter%204%20-%20Classification/Exercise%2042%20-%20Iris%20Classification%20Using%20a%20CART%20Decision%20Tree.ipynb (visited on 06/16/2022).

[35] DisplayR, *Machine Learning: Pruning Decision Trees*, en-US, Jul. 2017. [Online]. Available: https://www.displayr.com/machine-learning-pruning-decision-trees/ (visited on 06/05/2022).

[36] R. Singh, *A practical approach to Tree Pruning using sklearn Decision Trees*, en-us, Apr. 2020. [Online]. Available: https://ranvir.xyz/blog/practical-approach-to-tree-pruning-using-sklearn/ (visited on 06/16/2022).

[37] E. Krueger, *Pre-Pruning or Post-Pruning*, en, Jul. 2021. [Online]. Available: https://towardsdatascience.com/pre-pruning-or-post-pruning-1dbc8be5cb14 (visited on 06/16/2022).

[38] Scikit-:earn, *Post pruning decision trees with cost complexity pruning*, en, 2022. [Online]. Available: https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html (visited on 06/05/2022).

[39] Panjeh, *Scikit learn hyperparameter optimization for MLPClassifier*, en, Jun. 2020. [Online]. Available: https://panjeh.medium.com/scikit-learn-hyperparameter-optimization-for-mlpclassifier-4d670413042b (visited on 06/17/2022).

[40] R. Pramoditha, *Train a regression model using a decision tree*, en, Oct. 2020. [Online]. Available: https://towardsdatascience.com/train-a-regression-model-using-a-decision-tree-70012c22bcc1 (visited on 06/17/2022).