Neural Networks for the Age Classification of Abalone

Amanjit Gill

June 18, 2023

1 Introduction

Abalone are highly valued for their culinary and scientific uses. This has lead to their endangerment through overfishing and unsustainable farming practices (International Union for Conservation of Nature and Natural Resources, 2022). While some action has been taken to correct the course and restore regional populations (Duggan, 2018), one particular area of difficulty is that of identifying the age of an abalone. Farmers seek to avoid harvesting abalone that are young and have not yet developed sex characteristics, but present methods for determining the age and sex of an abalone are intrusive and time-consuming, causing stress that may result in stunted growth or death (Killoran, 2021).

Researchers and stakeholders have put considerable energy into developing less intrusive methods for identifying or otherwise managing farmed abalone. For example, Gurney, Mundy, and Porteus (2005) explored the use of oxygen isotopes on shells in order to reveal their age, and others have experimented with population seeding (Anderson, 2003) in order to encourage renewal. In addition, practitioners in computer science and statistics have been devoted to the problem of abalone characteristisation for some time. For instance, the use of linear regression has been extensively explored by Mehta (2019) under a variety of optimisation and regularisation regimes, yielding reasonable results. Guney et al. (2022) have investigated a number of supervised machine learning algorithms, such as KNN (K-nearest neighhours), decision trees, neutral networks and support vector machines, obtaining strong results in nearly all cases; by contrast, Buntarto (2021) attempted linear regression with target transformation and obtained poorer results. Another notable example is that of Sahin et al. (1995), who applied a number of measures to reduce overfitting in their neural network, yielding better results than were previously observed with a similar basic architecture.

The present study focuses solely on neural networks, serving as an introduction to this method's capabilities and challenges in the context of abalone classification. A number of configurations shall be presented and compared with the aid of computed metrics and visualisations, showcasing the potential utility of neural networks in addressing the aforementioned problems with determining the age of an abalone.

2 Methodology

2.1 Preprocessing

The dataset in use for this analysis is the well-known abalone dataset that is part of the UCI (University of California, Irvine) repository commonly used to source data for education in machine learning (Nash et al., 1995). This dataset consists of the following features that are characteristics or measurements taken from over 4000 abalone.

- Sex
- Length, diameter and height
- Whole, shucked, viscera and shell weights
- Number of rings on the shell

2.1.1 Numerical inputs

Before commencing model development, it is considered instructive to examine the distributions of the input features, which comprise all of the above-listed features except for Rings, which is a proxy for Age (the target feature). Figure 1 shows these distributions. Note that Sex has been omitted because it is nominal-categorical, not continuous (it will *not* be omitted from the model altogether).

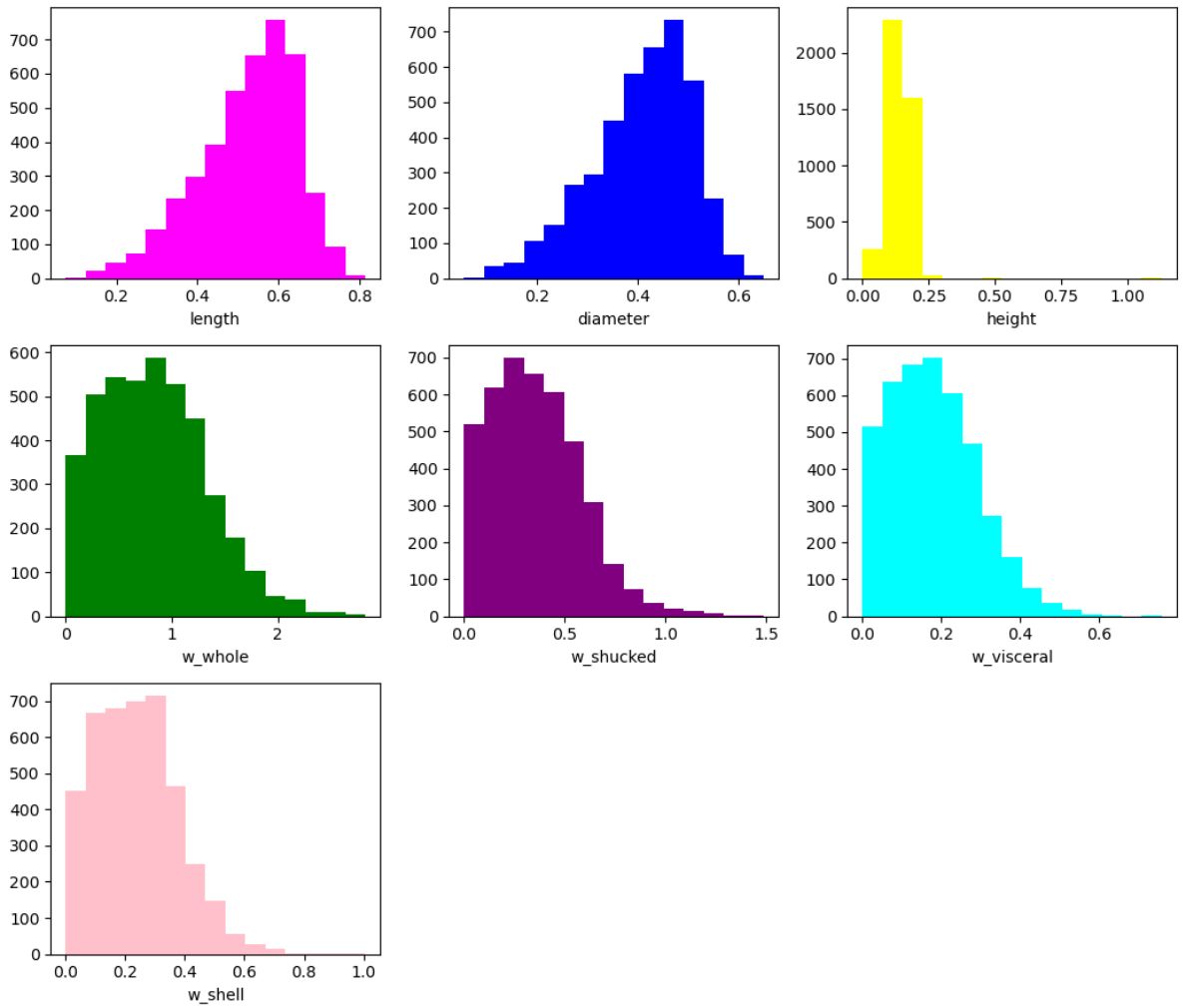

Figure 1: Distribution of Input Features.

Figure 1 shows that all of the numerical inputs are skewed, indicating the possible presence of outliers for some of the features. There are a number of methods for identifying and removing outliers (Sharma, 2018), but a simple method based on IQR (interquartile range) has been chosen, in which any point outside of the following interval is removed:

$$[Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR]$$

With outliers removed, the distributions of the numerical input features change to those shown in Figure 2.

Preprocessing of these numerical input features is now complete. Scaling is unnecessary, as these features have already been scaled at the source repository (Nash et al., 1995).

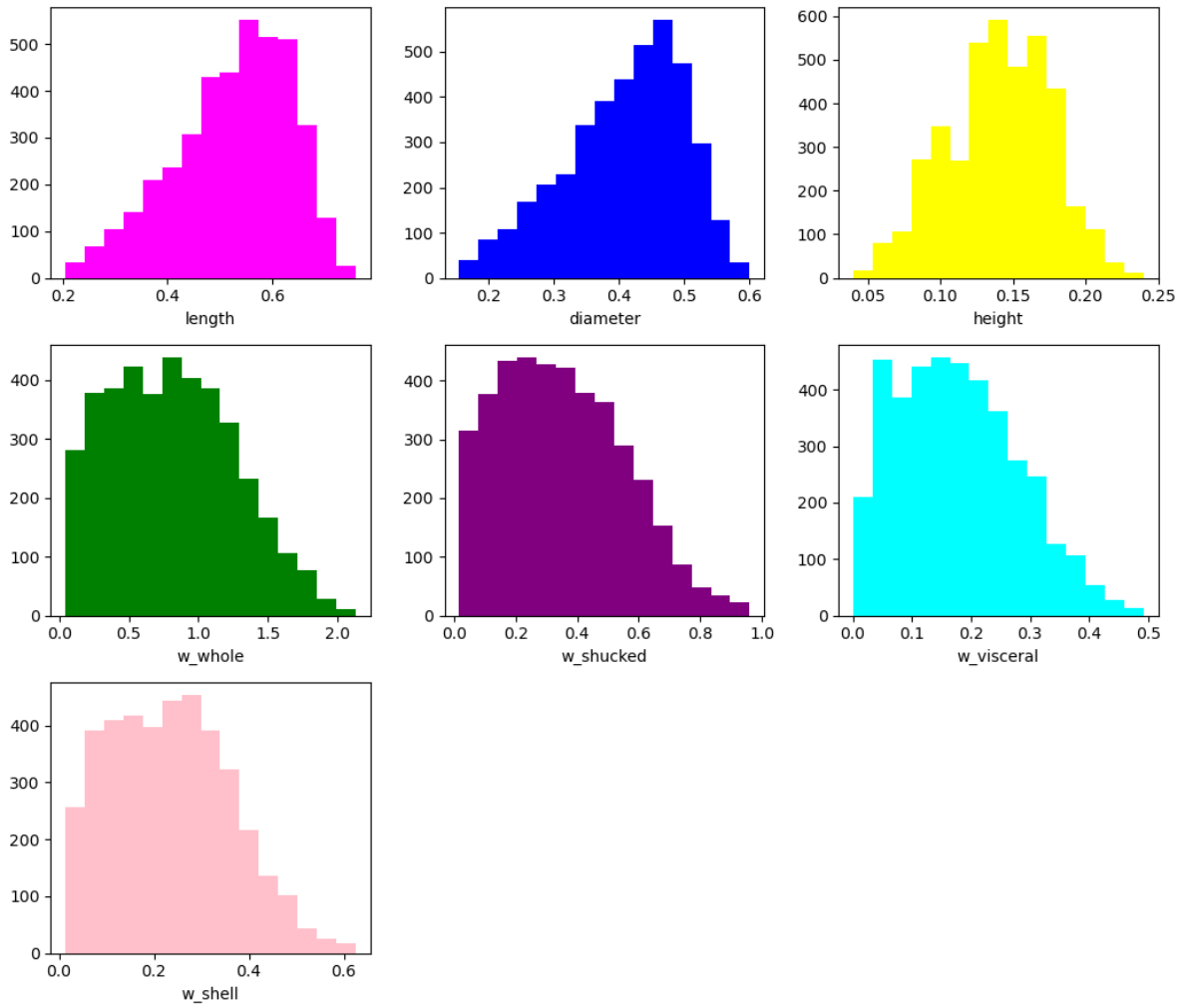

Figure 2: Distribution of Input Features.

2.1.2 Categorical input

The Sex feature must be encoded into numerals, to allow its inclusion into the classifier. To this end, the Sex column has been replaced by three columns through one-hot-encoding (Géron, 2019); one for Female, one for Infant, and one for Male. For any sample, only one of these three fields will be valued 1; the other two must be valued 0. An example is shown below.

F	I	M
0	0	1
1	0	0
1	0	0
0	0	1

Table 1: One-hot-encoding.

2.1.3 Target feature

Because the aim of the present exercise is to classify abalone by age, not number of rings, the Rings column is converted to Age by simply adding 1.5 to each value. This new Age column is then stripped of outliers, using the same method as that used for the numerical input features, after which it is converted into four classes according to the rules below:

- Class 1: Age 7 or below
- Class 2: Older than 7, up to (and including) 10
- Class 3: Older than 10, up to (and including) 15
- Class 4: Older than 15

It is evident from Figure 3 that the output is unbalanced. This may result in a classifier that is inaccurate for the underrepresented classes (Géron, 2019), which should be considered when evaluating performance.

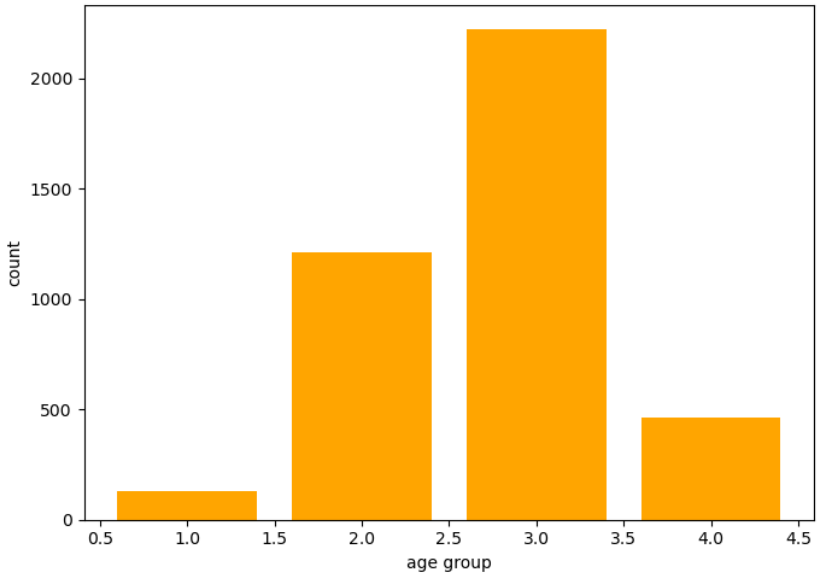

Figure 3: Distribution of Input Features.

2.2 Investigation

In order to assess the effects of different hyperparameter settings, a neural network (in the form of a multilayer percepton classifier) has been repeatedly trained by holding its basic configuration steady while altering a hyperparameter or other setting of interest. These settings of interest are:

- Number of neurons in a single hidden layer
- Learning rate
- Number of hidden layers

- Type of optimiser (SGD or Adam)

At the outset, the preprocessed abalone dataset has been split into two subsets, one for training the classifier, and the other for testing it. This split has remained constant throughout all training runs - it has not changed throughout the investigation, in order to ensure that any observed changes are due to an altered configuration and not due to sensitivity to variations in the data.

For the (hyper)parameters of interest, the following procedure has been undertaken:

1. The classifier is trained n times per hidden neurons setting:
   - n times with 5 hidden neurons, then mean accuracy is calculated
   - n times with 10 hidden neurons, then ...
   - n times with 15 hidden neurons, then ...
   - n times with 20 hidden neurons, then ...
2. The classifier is re-initialised with the best number of neurons, then trained n times per learning rate setting:
   - n times with learning rate = 0.1, then mean accuracy is calculated
   - n times with learning rate = 0.05, then ...
   - n times with learning rate = 0.01, then ...
   - n times with learning rate = 0.005, then ...
   - n times with learning rate = 0.001, then ...
   - n times with learning rate = 0.0005, then ...
   - n times with learning rate = 0.0001, then ...
3. The classifier is re-initialised with the best number of neurons and the best learning rate, then trained n times per layers setting:
   - n times with 1 hidden layer, then mean accuracy is calculated
   - n times with 2 hidden layers, then ...
   - n times with 3 hidden layers, then ...
4. The classifier is re-initialised with the best number of neurons, the best learning rate, and the best number of hidden layers, then trained n times per solver:
   - n times with solver = 'sgd', then mean accuracy is calculated
   - n times with solver = 'adam', then ...

For each set of n training runs, the classifier starts each training run with different initial weights and biases. Repeating the use of each setting n times with different starting conditions helps to guard against the algorithm finding a suboptimal local solution, and also gives some assurance that the returned solution is not sensitive to the starting position (Robert Keim, 2020).

Within each training run, the number of epochs is also configurable; the chosen value is 1000; this value would be unnecessarily high for some combinations of hyperparameters,

but insufficient for convergence for others. However, it cannot be increased any more than this due to hardware constraints.

For clarity, the above procedure omits a few important computations. However, it should be noted that for each set of n training runs, the following metrics are calculated in addition to the mean accuracy:

- standard deviation of accuracy (for both the training and testing set)
- 95% confidence interval of accuracy (for only the testing set)

The confidence interval is calculated using the method outlined by (Sebastian Raschka, 2022); it matches the common formula for the confidence interval for a variable that follows a Gaussian distribution, but it uses Student-t to account for the relatively small number of trials (for the present analysis, n=10 to save computation time during the investigation phase, then it is increased to 20 for the final phase in which the metrics for the best model are calculated):

$$CI = \overline{acc} \pm t \times SE$$
$$SE = \frac{SD}{\sqrt{n}}$$
$$where\ n = no.\ trials$$

In addition, the Python source code has functions for the following visual metrics; these are called whenever further examination of a result is required:

- loss curve for analysing how the training loss changes as epochs progress
- multiclass ROC (receiver operator characteristic) curve, showing a one-versus-rest curve for each class

3 Results and Discussion

As aforementioned, the number of trials, n, per setting has been set to 10. This means that every trialled hyperparameter value has been used to train the classifier 10 times, and then the accuracy score from the 10 trials has been used to compute the mean, standard deviation and 95% confidence interval of the *accuracy* for the model - but only for the hyperparameter value of interest.

3.1 Number of hidden neurons

The text output of the model, for each trialled number of hidden neurons, is given below.

	mean_train	mean_test	sd_train	sd_test	CI_test
5	0.7119	0.7139	0.0036	0.0036	0.7113 - 0.7165
10	0.7176	0.713	0.0033	0.005	0.7094 - 0.7166
15	0.7174	0.713	0.0045	0.0037	0.7104 - 0.7156
20	0.719	0.7151	0.0021	0.0026	0.7132 - 0.717

Examining the mean accuracy when the model is asked to classify the test subset reveals that the best number of neurons for the hidden layer is 20. However, it must be observed that the performance of the model for all of the trialled values is very similar.

3.2 Learning rate

The text output of the model, for each trialled learning rate, is given below.

	mean_train	mean_test	sd_train	sd_test	CI_test
0.1	0.6937	0.6899	0.0388	0.039	0.662 - 0.7178
0.05	0.7166	0.7124	0.0046	0.0059	0.7082 - 0.7166
0.01	0.719	0.7151	0.0021	0.0026	0.7132 - 0.717
0.005	0.7175	0.7146	0.0023	0.0029	0.7125 - 0.7167
0.001	0.6847	0.676	0.0027	0.0014	0.675 - 0.677
0.0005	0.6779	0.6742	0.0022	0.0016	0.6731 - 0.6753
0.0001	0.666	0.6756	0.009	0.0039	0.6728 - 0.6784

According to the mean test accuracy, the best learning rate, when the classifier has a hidden layer containing 20 neurons, is 0.01. This is a good result, as training tends to occur rapidly at this learning rate. Notably, the lower learning rates perform worse - and a few of them were unable to converge within 1000 epochs.

3.3 Number of hidden layers

The text output of the model, for each trialled number of hidden layers, is given below.

	mean_train	mean_test	sd_train	sd_test	CI_test
1	0.719	0.7151	0.0021	0.0026	0.7132 - 0.717
2	0.7194	0.71	0.0042	0.0058	0.7059 - 0.7141
3	0.7123	0.7013	0.0093	0.0134	0.6917 - 0.7109

The results for one, two and three hidden layers reveal that a single hidden layer performs marginally better than two or three, rendering additional complexity unnecessary. In fact, the additional layers appear to make the results slightly worse.

3.4 Solver

The text output of the model, for each trialled solver, is given below.

	mean_train	mean_test	sd_train	sd_test	CI_test
sgd	0.719	0.7151	0.0021	0.0026	0.7132 - 0.717
adam	0.7204	0.7145	0.0031	0.0063	0.71 - 0.719

SGD (stochastic gradient descent) is the solution algorithm that has been applied to all the previous trials in this study. The above comparison with Adam (adaptive momentum) suggests that there is no improvement to be gained - for this particular dataset - when Adam is the selected solver. Therefore, the outcome of this part of the investigation is that the originally chosen solver, SGD, should remain the one in use.

No. hidden neurons in one layer	20
Learning rate	0.01
No. hidden layers	1
Solver	SGD

Table 2: Configuration for best classifier.

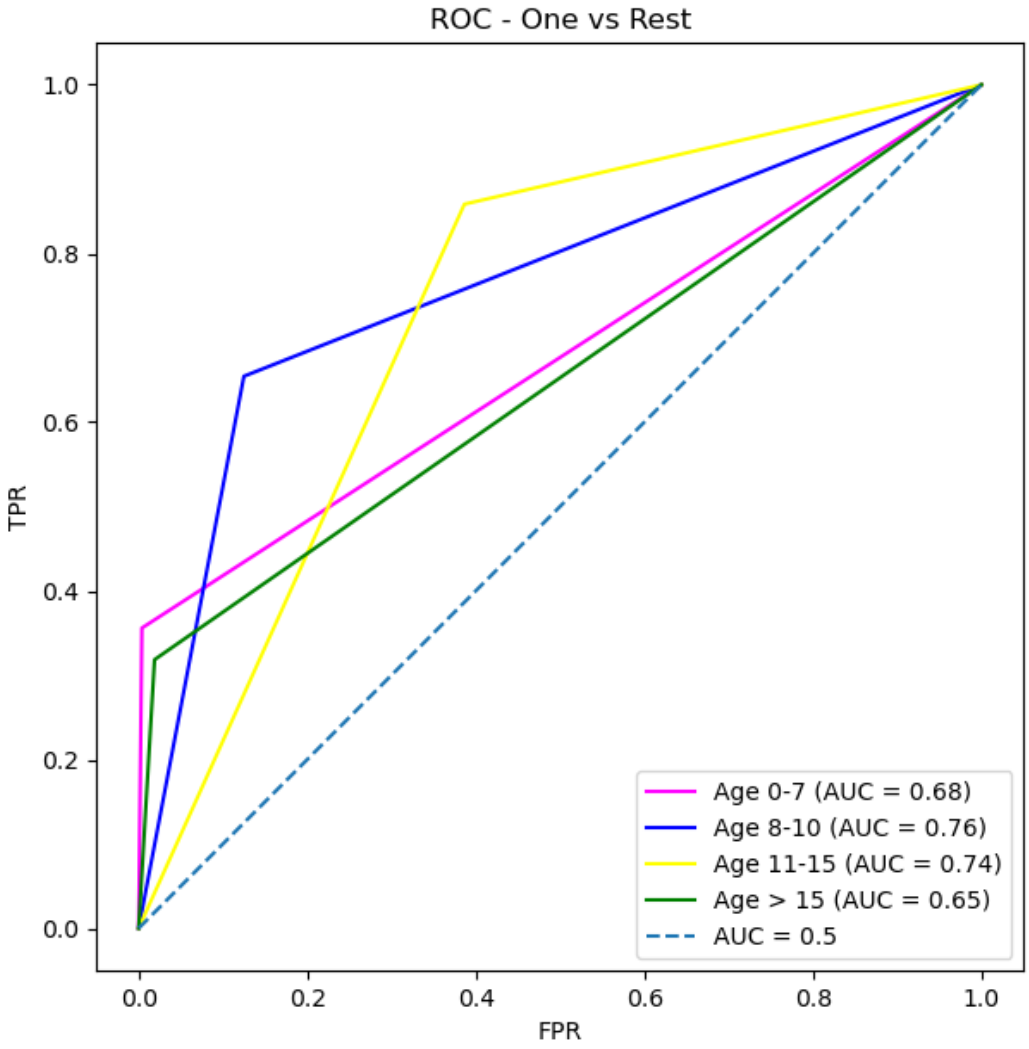

Figure 4: Confusion matrix for selected MLP classifier.

3.5 Best model

The best model for classifying abalone has a configuration as summarised in Table 2, and its one-versus-the-rest ROC curve is shown in Figure 4.

A confusion matrix for this model is given in Figure 5. This reveals that as expected, the unbalanced dataset has likely contributed to very poor performance for the two underrepresented classes, 1 (age <= 7) and 4 (age > 15). The most prevalent class, 3 (age

11 to 15), is unsurprisingly the easiest for the neural network to classify; it does this correctly 86% of the time. The accuracy of this model, by class, is given in Table 3.

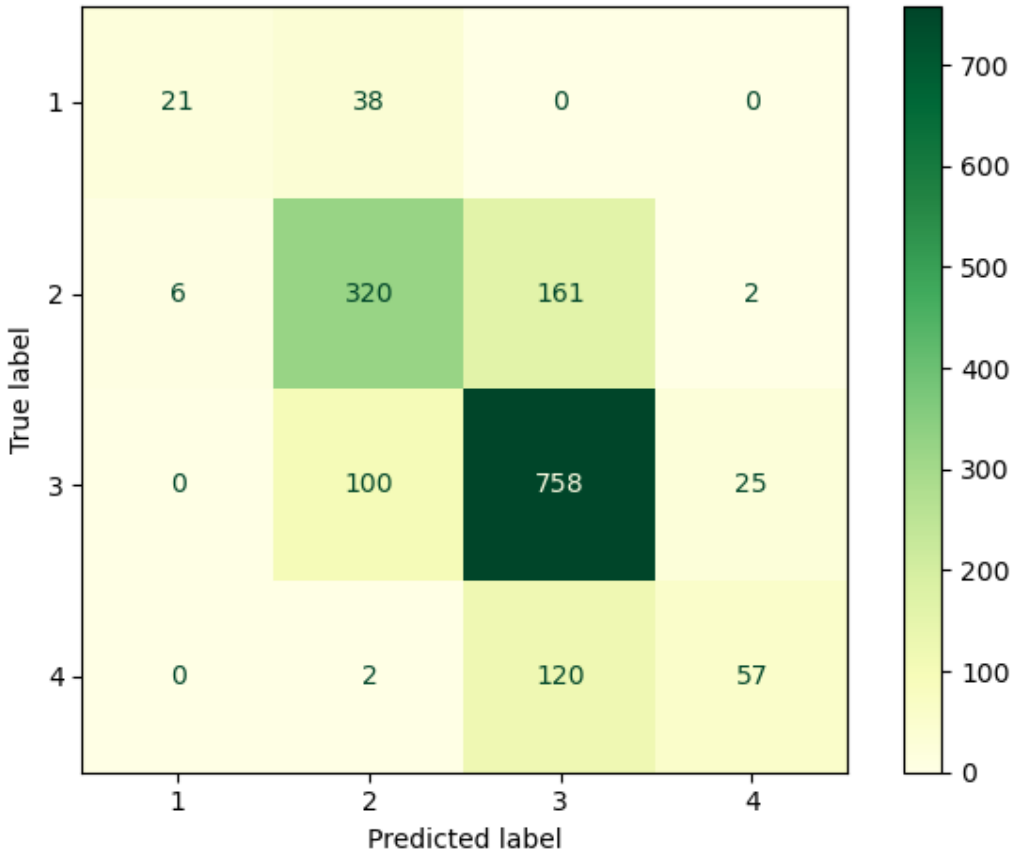

Figure 5: Confusion matrix for selected MLP classifier.

Class	Age Group	Accuracy %
1	up to 7	36
2	up to 10	65
3	up to 15	86
4	greater than 15	32

Table 3: Accuracy by class.

In addition to the lack of representation within the dataset, it is possible that the two classes for which the model underperforms are not easily separable from the others. If it is the case that these classes have similar values for some input features to the most prevalent classes, then the classifier may not be able to separate abalone belonging to the two classes, forcing it to assign abalone of class 1 to class 3 (simply because class 3 seems more probable).

4 Conclusion

The aim of this study, which is to showcase the applications that neural networks might have in being part of the solution to a long-standing problem with abalone classification,

has largely been met. It is, however, noted that the final classifier model, as it currently stands, would not be suitable for determining the age of an abalone, because it appears to be incapable of reliably classifying very young and very old abalone. Therefore, more investigation is warranted as to how this shortcoming might be mitigated.

References

Anderson, Genny (2003). *Abalone History and Future*. https://www.marinebio.net/marinescience/06future/abhist.htm. [Accessed on 15-06-2023].

Buntarto, Dimas (2021). *Effect of Target Transformation on Abalone Age Prediction Using Regression Model*. https://medium.com/analytics-vidhya/effect-of-target-transformation-on-abalone-age-prediction-using-regression-model-da6d3a185a31. [Accessed on 16-06-2023].

Duggan, Tara (2018). *California abalone season sunk until 2021 to give stressed population time to rebuild*. https://www.sfchronicle.com/food/article/No-abalone-diving-allowed-in-California-until-2021-13460882.php. [Accessed on 15-06-2023].

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. OReilly Media. ISBN: 9781492032618.

Guney, Seda et al. (2022). "Abalone Age Prediction Using Machine Learning". In: *Pattern Recognition and Artificial Intelligence*. Ed. by Chawki Djeddi et al. Cham: Springer International Publishing, pp. 329–338. ISBN: 978-3-031-04112-9.

Gurney, L.J., C. Mundy, and M.C. Porteus (2005). "Determining age and growth of abalone using stable oxygen isotopes: a tool for fisheries management". In: *Fisheries Research* 72.2, pp. 353–360. ISSN: 0165-7836. DOI: https://doi.org/10.1016/j.fishres.2004.12.001. URL: https://www.sciencedirect.com/science/article/pii/S0165783604003054.

International Union for Conservation of Nature and Natural Resources (2022). *Human activity devastating marine species from mammals to corals - IUCN Red List*. https://www.iucn.org/press-release/202212/human-activity-devastating-marine-species-mammals-corals-iucn-red-list. [Accessed on 15-06-2023].

Killoran, Tianna (2021). *The future of abalone farming*. https://www.jcu.edu.au/this-is-uni/science-and-technology/articles/the-future-of-abalone-farming. [Accessed on 15-06-2023].

Mehta, Kunj (2019). *Abalone Age Prediction Problem: A Review*. https://www.ijcaonline.org/archives/volume178/number50/mehta-2019-ijca-919425.pdf. [Accessed on 16-06-2023].

Nash, Warwick et al. (1995). *Abalone*. UCI Machine Learning Repository.

Robert Keim (2020). *Understanding Local Minima in Neural Network Training*. [Accessed on 17-06-2023].

Sahin, Egemen et al. (Nov. 2018). "Abalone Life Phase Classification with Deep Learning". In: pp. 163–167. DOI: 10.1109/ISCMI.2018.8703232.

Sebastian Raschka (2022). *Creating Confidence Intervals for Machine Learning Classifiers*. https://sebastianraschka.com/blog/2022/confidence-intervals-for-ml.html#method-4-confidence-intervals-from-retraining-models-with-different-random-seeds. [Accessed on 16-06-2023].

Sharma, Natasha (2018). *Ways to Detect and Remove the Outliers*. https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba. [Accessed on 16-06-2023].