AGE CLASSIFICATION OF ABALONE

USING FULLY CONNECTED NEURAL NETWORKS

A MACHINE LEARNING APPROACH

Author	Anirban Chakrabarty
zID	z5626947
Course	ZZSC5836 Data Mining and Machine Learning (H225 Online)
Tutorial Time	March - April 2025
Tutor	Dr Sarat Moka
Date	20/April/2025

Contents

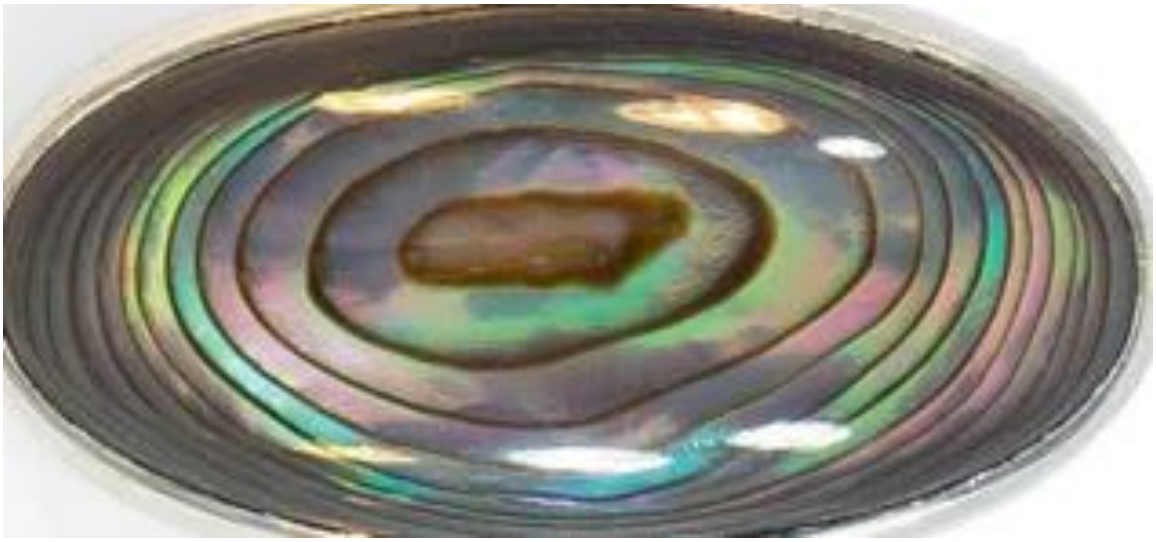
1. Abstract

Predicting the age of abalones is traditionally a labor-intensive task. Manual age estimation involves physical cutting through shells and counting growth rings under a microscope.

This project explores the application of Fully Connected Neural Networks (FCNNs) to predict abalone age classes using physical measurements, aiming to automate and accelerate this process. The dataset is sourced from the UCI Machine Learning Repository and contains multiple biometric features. The target variable, "ring age," is categorized into four classes.

We experiment with various FCNN configurations/ hyper parameters — different numbers of hidden layers, neurons, learning rates, and optimizers (SGD and Adam) — to identify the optimal model. The model performance is evaluated through accuracy, confusion matrix, and ROC/AUC curves.

Our optimal model uses Adam optimizer, 2 hidden layers with 15 neurons each, and a learning rate of 0.001, achieving a test accuracy of 72.484%.



2. Introduction

The abalone is a type of mollusk whose age is crucial for ecological and commercial purposes. Traditionally, age is estimated by slicing the shell and counting growth rings, a method that is both invasive and time-consuming. The present study explores a machine learning-based approach to predict abalone age using non-invasive measurements such as length, diameter, and various weight parameters.

Ring-age is calculated in years as number of rings +1.5. We employ a classification approach by segmenting the ring-age into four categories or classes.

- Class 0: Age <= 7
- Class 1: 8 <= Age <= 10
- Class 2: 11 <= Age <= 15
- Class 3: Age > 15

This project utilizes Fully Connected Neural Networks (FCNNs) to classify the abalone into these categories based on their physical features. The key goal is to evaluate how various neural network parameters influence classification performance and to identify the most effective model.

3. Literature Review

The Abalone dataset has been widely used in regression and classification problems, particularly in age prediction tasks. Traditional machine learning methods like Decision Trees, Support Vector Machines, and k-NN have been tested, but recent advancements in deep learning offer new opportunities for more accurate and scalable models. As highlighted by Goodfellow et al. (2016), deep learning models like FCNNs can learn highly non-linear relationships in structured data.

Neural networks, especially FCNNs, have demonstrated robust performance in classification tasks. Prior research shows that tuning the number of hidden layers and neurons, learning rates, and optimizer algorithms can significantly affect performance. Metrics such as confusion matrices and ROC/AUC scores, commonly recommended in classification tasks (Goodfellow et al., 2016), were employed for evaluating performance, especially in imbalanced multi-class scenarios.

4. Methodology

4.1 Data Preprocessing

The dataset was sourced from the UCI Machine Learning Repository (Dua & Graff, 2019). It contains 4,177 rows and 8 input features like Length, Diameter, Height, Whole Weight, Shucked Weight, Viscera Weight, Shell Weight and Sex. The target / response variable 'Rings' (integer), when added 1.5 to it, gives the age in years (ring-age).

The "Sex" column was transformed using One Hot Encoding. The "ring age" numeric column was converted into four categorical classes, described above.

4.2 Feature Overview

The dataset contains 4,177 rows and 8 input features:

- Length, Diameter, Height
- Whole Weight, Shucked Weight, Viscera Weight, Shell Weight
- Sex is a categorical feature that has been one-hot-encoded into 3 fields, Sex_M for male, Sex_F for female and Sex_I for infants.

4.3 Model Architecture

The fully connected neural network (FCNN) model comprises:

- An input layer matching the number of features
- One to three hidden layers with varying neurons (5, 10, 15, 20)
- Output layer with 4 neurons and softmax activation for multi-class classification
- Optimizer: SGD or Adam
- Loss Function: Sparse Categorical Crossentropy
- Metrics: Accuracy, Confusion Matrix, ROC/AUC

The FCNN was built using Keras (Chollet, 2015), with TensorFlow (Abadi et al., 2016) as the computational backend.

4.4 Evaluation Strategy

We evaluate models based on their test-set accuracy. Further evaluation is performed using confusion matrices and ROC/AUC curves to assess class-wise performance.

5. Analysis and Experiments

5.1 Step 1: Exploratory Data Analysis

We examined the distribution of age classes and correlations between features using heat-maps and histograms. The data showed moderate correlations between weight-related features and age classes.

5.1.1 Histogram of Abalones by RingAgeClass

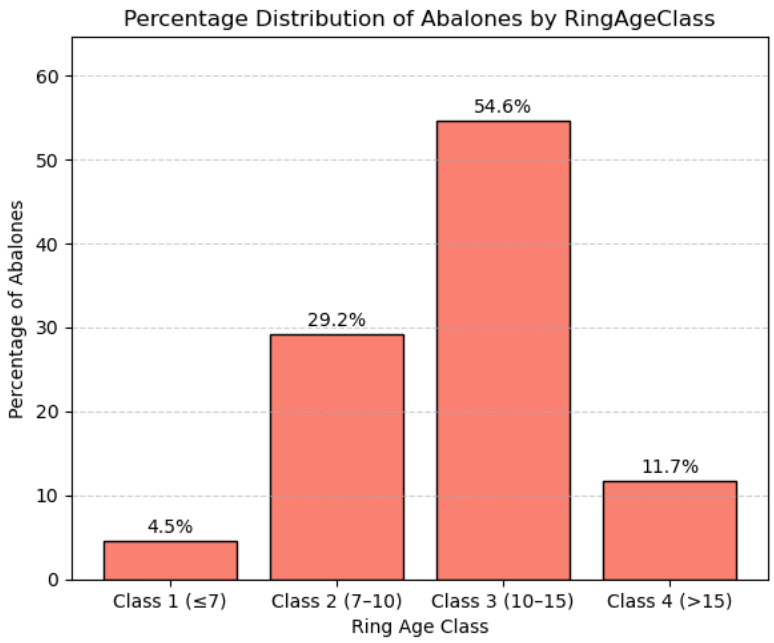


Fig1: Percentage Distribution of Abalones by RingAgeClass

The above histogram, titled **"Percentage Distribution of Abalones by RingAgeClass,"** visually represents the relative frequencies of abalone samples falling into each of the four ring-age-based classes. These classes are defined based on estimated biological age using the formula:

Age = Number of Rings + 1.5 years

Accordingly, the dataset has been categorized into four classes for classification purposes:

- **Class 0**: Age ≤ 7 years
- **Class 1**: 8 ≤ Age ≤ 10 years
- **Class 2**: 11 ≤ Age ≤ 15 years
- **Class 3**: Age > 15 years

5.1.2 Interpretation of Ring-Age-Class Histogram:

- **Class 0 (≤7 years)**:
  - Represents only 4.5% of the total abalone population.
  - This is the smallest age group in the dataset.
  - It indicates that young abalones are relatively rare in the sample.
- **Class 1 (8–10 years)**:
  - Comprises about 29.2% of the data.
  - These are mid-aged abalones and are more commonly observed.
- **Class 2 (11–15 years)**:
  - Dominates the dataset with 54.6% representation.
  - This is the most prevalent age group, forming over half the population.
  - Suggests that the majority of abalones sampled are middle-aged to older.
- **Class 3 (>15 years)**:
  - Accounts for **11.7%** of the sample.
  - These are the oldest abalones in the data.
  - Less frequent but still a significant minority.

5.1.3 Significance Drawn from Histogram

This class imbalance is important for training the classifier. The dominance of Class 2 might bias the model towards favoring predictions in that category. Proper evaluation methods (e.g., confusion matrices and ROC/AUC curves) must be employed to ensure fair performance across all classes. Additionally, techniques like class weighting or resampling might be required during model training to address this imbalance.

5.1.4 Correlation Heatmap of Physical Features & Ring-Age of Abalones

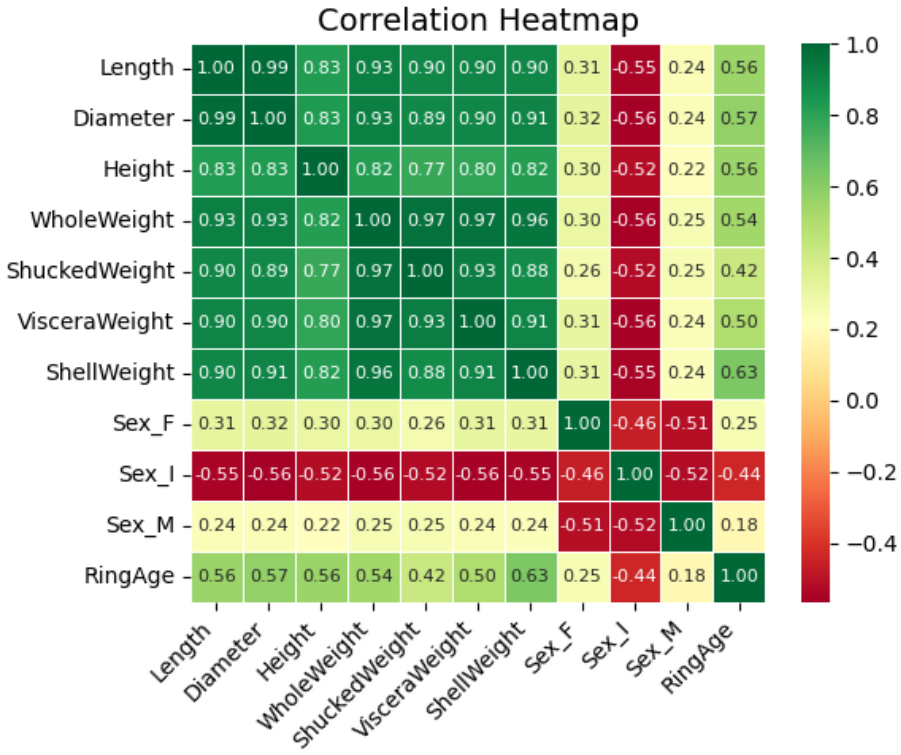


Fig2: Correlation Heatmap of Physical features and Ring-Age of Abalones

The correlation heatmap visually displays how strongly each feature of abalones is linearly related to the target variable RingAge, which is often used as a proxy for predicting the age of abalones. Here's a breakdown of the significance and key inferences from the heatmap.

5.1.5 Key Inferences Regarding Age Prediction (RingAge)

Most Positively Correlated Features: These features have higher correlation values with RingAge, indicating they are potentially good predictors:

Feature	Correlation with RingAge
ShellWeight	0.63
WholeWeight	0.54
Length	0.56
Diameter	0.57
Height	0.56

- **ShellWeight (0.63)** is the most strongly correlated feature with RingAge, implying that as abalones grow older, their shell weight increases significantly.
- **WholeWeight, Length, Diameter,** and **Height** also show moderate positive correlations, reinforcing the idea that larger, heavier abalones tend to be older.

Weakly Correlated Features: following featues have moderate correlations but might still contribute when combined with others in a multivariate model

Feature	Correlation
ShuckedWeight	0.42
VisceraWeight	0.44

Low or Negative Correlation Features

Feature	Correlation
Sex_M	0.18
Sex_F	0.25
Sex_I	-0.44

- **Sex_I** (Infant) has a moderate negative correlation with age, which makes sense — younger abalones are more likely to be labeled as 'Infant'.
- **Sex_F** and **Sex_M** (male/ female) have weak correlations, suggesting that gender has limited predictive power on its own for age.

Implications for Modeling

1. **Best Predictors**: Focus on **ShellWeight**, **Length**, **Diameter**, and **WholeWeight** when building models.
2. **Dimensionality Reduction**: Features like Sex might not be essential unless you're exploring interaction terms.
3. **Multicollinearity**: Features like Length, Diameter, and Height are highly correlated with each other (e.g., Length–Diameter: 0.99). Therefore, techniques like PCA or regularization (e.g., Lasso) may be used to reduce redundancy.

5.2 Step 2: Tuning Hidden Layer Neurons for SGD Optimizer

First, we investigate how varying the number of neurons in the hidden layer of a fully connected neural network affects the classification accuracy for predicting abalone age classes. We develop a dense neural network with one hidden layer and vary the *number of hidden neurons* to be 5, 10, 15 and 20 in order to investigate the performance of the model using Stochastic Gradient Descent (SGD) optimizer at first. We determine the optimal number of neurons in the hidden layer from the range of values considered.

5.2.1 Experimental Setup

- Model Architecture: A simple feedforward neural network with:
  - One input layer (with features derived from abalone measurements),
  - One hidden layer
  - One output layer with softmax activation for multiclass classification.
- Hyperparameter Tuned: Number of neurons in the hidden layer.
- Repetitions of experiment: 10 runs with different initial weights
- Metric Evaluated: Mean Accuracy on test set with 95% Confidence Intervals.
- Fixed Parameters: Optimizer (SGD), learning rate (0.01), batch size (32) and number of epochs (100) are held constant during this experiment.

5.2.2 Observations

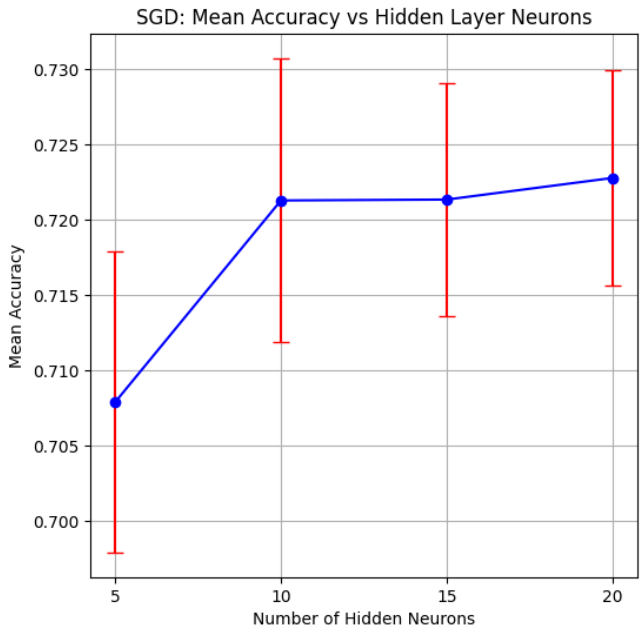


Fig3: Number of Hidden Layer Neurons vs Accuracy in FCNN with 1 hidden layer & SGD optimizer

Using a single hidden layer and SGD, we varied the number of hidden layer neurons and calculated the resulting Mean Accuracies and 95% Confidence Intervals (CI):

- 5 Neurons → Mean Accuracy: 0.70790 → 95% CI: (0.69790, 0.71790)
- 10 Neurons → Mean Accuracy: 0.72130 → 95% CI: (0.71187, 0.73074)
- 15 Neurons → Mean Accuracy: 0.72136 → 95% CI: (0.71364, 0.72908)
- 20 Neurons → Mean Accuracy: 0.72280 → 95% CI: (0.71563, 0.72997)

Best number of hidden neurons = 20

5.2.3 Trends in Observation

As the number of hidden neurons in the single-layer FCNN increased from 5 to 20, we observed a general trend of improvement in the mean accuracy achieved by the model when using the Stochastic Gradient Descent (SGD) optimizer. The mean accuracy started at approximately 0.70790 with 5 neurons and progressively increased to 0.72130 with 10 neurons, 0.72136 with 15 neurons, and finally reached 0.72280 with 20 neurons.

The 95% confidence intervals also provide insights into the variability of the model's performance. As the number of neurons increased, the confidence intervals appeared to narrow slightly, suggesting a potentially more stable performance, although the overlap between the intervals indicates that the differences in mean accuracy might not be statistically significant across all tested neuron counts. Notably, the increase in mean accuracy between 10 and 15 neurons was marginal, and a further increase to 20 neurons yielded a slightly more noticeable improvement, albeit still relatively small.

5.2.4 Explanation of Trends

The observed trend of increasing accuracy with a greater number of hidden neurons can be attributed to the increased capacity of the neural network to learn and represent complex relationships within the abalone dataset. With more neurons in the hidden layer, the network has more parameters, allowing it to model more intricate non-linear decision boundaries necessary for classifying abalone age based on the provided physical measurements.

The initial jump in performance from 5 to 10 neurons suggests that a minimal number of neurons might limit the network's ability to capture the underlying patterns in the data, leading to underfitting. As the number of neurons increases, the network gains the representational power to better approximate the complex function mapping the input features to the age classes.

The diminishing returns observed beyond 10 neurons, where the increase in accuracy becomes smaller with each increment in neuron count, could indicate that the model is approaching a point of diminishing returns for this specific network architecture and optimizer. It's possible that the complexity of the relationships in the data that can be effectively learned with a single hidden layer is being saturated. Alternatively, further improvements might require adjustments to other hyperparameters or a more complex network architecture, such as adding more hidden layers.

The slight narrowing of the confidence intervals with increasing neuron count could suggest that a larger network is less sensitive to the random initialization of weights and biases, leading to more consistent performance across different training runs.

5.2.5 Conclusion from Hidden Neurons Hyper Parameter Tuning

Based on the experimental results using a single hidden layer and the Stochastic Gradient Descent (SGD) optimizer, increasing the number of hidden neurons generally led to improved classification accuracy for predicting abalone age classes. The highest mean accuracy of approximately 0.72280 was achieved with 20 hidden neurons.

While the increase in performance beyond 10 neurons was marginal, the trend suggests that a greater number of neurons allows the model to better capture the complexity of the data. However, the relatively small improvements beyond a certain point also indicate that simply increasing the number of neurons in a single hidden layer might not be sufficient to achieve significantly higher accuracy.

Therefore, for a single hidden layer FCNN using SGD with the fixed learning rate, batch size, and number of epochs considered in this experiment, 20 hidden neurons appeared to be the most effective configuration. However, it is important to note that this conclusion is specific to the tested range and the fixed hyperparameters. Further investigation into other hyperparameters, such as the learning rate, or exploring deeper network architectures with multiple hidden layers, might yield even better performance. The results from this experiment provide a valuable starting point for subsequent hyperparameter tuning stages.

5.3 Step 3: Learning-Rate Hyper Parameter Tuning for SGD

To identify the optimal learning rate, we conducted a series of experiments using a Single Hidden Layer Neural Network with 20 neurons from previous experiment's best performing architecture, employing the Stochastic Gradient Descent (SGD) optimizer. The learning rate is a critical hyperparameter that governs how quickly the network updates its weights during training. An appropriate learning rate ensures stable convergence and prevents issues like overshooting minima or excessively slow training.

5.3.1 Experiment Setup

- Architecture: 20 Neurons in 1 hidden layer as per best performance from previous experiment.
- Optimizer: Stochastic Gradient Descent (SGD)
- Learning Rates Tested: 0.0001, 0.001, 0.01, 0.1
- Evaluation Metric: Mean Validation Accuracy on test set with 95% CI

5.3.2 Observations

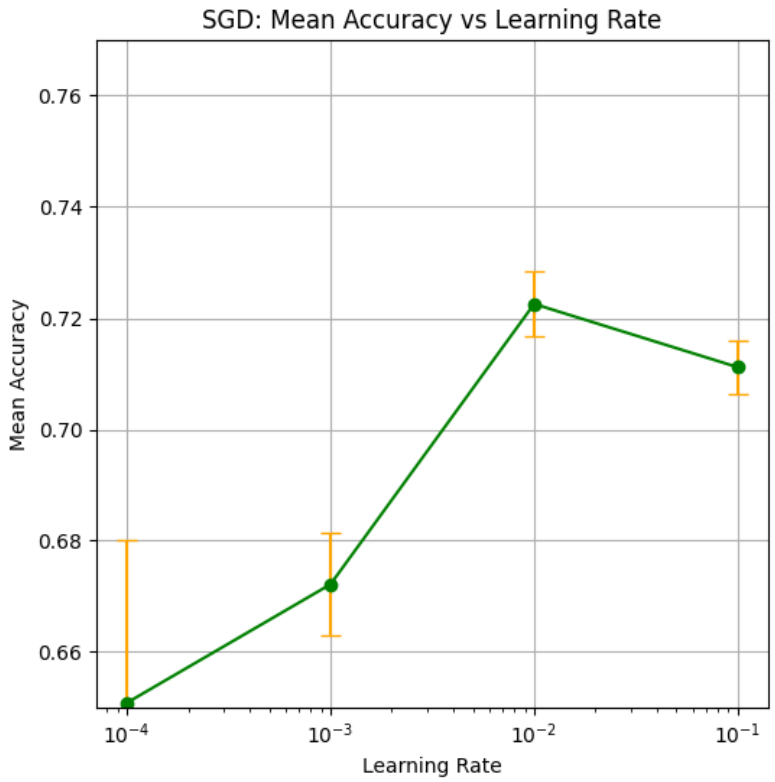


Fig4: Learning Rate vs Accuracy in FCNN with 1 hidden layer using 20 neurons & SGD optimizer

Using 20 neurons, in a single hidden layer & SGD optimizer from the previous experiment, we varied Learning Rate and calculated the resulting Mean Accuracies and 95% Confidence Intervals (CI):

- Learning Rate: 0.1 → Mean Accuracy: 0.71119 → 95% CI: (0.70630, 0.71608)
- Learning Rate: 0.01 → Mean Accuracy: 0.72256 → 95% CI: (0.71667, 0.72845)
- Learning Rate: 0.001 → Mean Accuracy: 0.67217 → 95% CI: (0.66292, 0.68143)
- Learning Rate: 0.0001 → Mean Accuracy: 0.65087 → 95% CI: (0.62153, 0.68021)

Best Learning Rate: 0.01

5.3.3 Trends in Observation

The experiment varying the learning rate for a single hidden layer FCNN with 20 neurons and the SGD optimizer revealed a non-monotonic relationship between the learning rate and the mean accuracy. As learning rate increased from a very small value of 0.0001 to 0.01, the mean accuracy of the model showed a significant improvement, rising from approximately 0.65087 to the highest observed value of 0.72256. However, further increasing learning rate to 0.1 resulted in a decrease in mean accuracy to around 0.71119.

The 95% confidence intervals also varied across the tested learning rates. The widest confidence interval was observed at the lowest learning rate (0.0001), suggesting higher variability in the model's performance. As the learning rate increased to 0.01, the confidence interval narrowed, indicating more stable and consistent performance. Increasing the learning rate further to 0.1 resulted in a slightly wider confidence interval compared to 0.01, suggesting a potential increase in performance variability again.

5.3.4 Explanation of Trends

The observed trends highlight the crucial role of the learning rate in the training process of a neural network.

A very low learning rate (0.0001) leads to slow convergence. The weight updates in each iteration are very small, causing the model to take a long time to reach an optimal or even a reasonably good solution. The wide confidence interval at this learning rate might indicate that the training process is highly sensitive to the initial weight values and might easily get stuck in suboptimal local minima, leading to variable performance across different training runs.

Increasing the learning rate to an optimal value (0.01) allows the model to make more significant weight adjustments in each step, leading to faster convergence towards a minimum of the loss function. The highest mean accuracy and the relatively narrow

confidence interval at this learning rate suggest that it strikes a good balance between the speed of learning and the stability of the training process, allowing the model to effectively learn the underlying patterns in the data without overshooting the optimal parameter values.

However, using a too high learning rate (0.1) can cause the model to overshoot the minima of the loss function. The large weight updates can lead to oscillations around the optimal values or even divergence, preventing the model from converging to a good solution. The decrease in mean accuracy and the slightly wider confidence interval compared to the optimal learning rate indicate that the training process becomes less stable and the model's performance deteriorates.

The results underscore the importance of finding an appropriate learning rate that allows for efficient and stable training of the neural network.

5.3.5 Conclusion from Learning Rate Tuning:

The experiments on tuning the learning rate for a single hidden layer FCNN with 20 neurons and the SGD optimizer revealed that the optimal learning rate among the tested values was 0.01, which yielded the highest mean accuracy and a relatively stable performance as indicated by the narrow confidence interval.

A learning rate that is too low (0.0001) results in slow learning and potentially suboptimal performance with high variability. Conversely, a learning rate that is too high (0.1) can lead to instability in the training process and a decrease in model accuracy.

Therefore, for the given network architecture (single hidden layer with 20 neurons) and the SGD optimizer, a learning rate of 0.01 is identified as the most suitable among the tested values. This finding will be crucial for the subsequent steps of hyperparameter tuning and model development, where this optimal learning rate can be used as a fixed parameter while exploring other aspects of the model architecture or different optimizers.

5.4 Step 4: Number of Hidden Layers Tuning for SGD

To further optimize the performance of our Fully Connected Neural Network (FCNN) model, we investigated the impact of **varying the number of hidden layers** while keeping other key hyperparameters constant. This analysis helps identify the network depth that best captures the complexity of the input data without leading to overfitting or underfitting.

5.4.1 Experiment Configuration

- Neurons per Layer: 20 (best from previous tuning)
- Learning Rate: 0.01 (best from previous tuning)
- Optimizer: Stochastic Gradient Descent (SGD)
- Hidden Layers Tested: 1, 2, and 3
- Evaluation Metric: Mean Validation Accuracy on test set with 95% CI

5.4.2 Observations

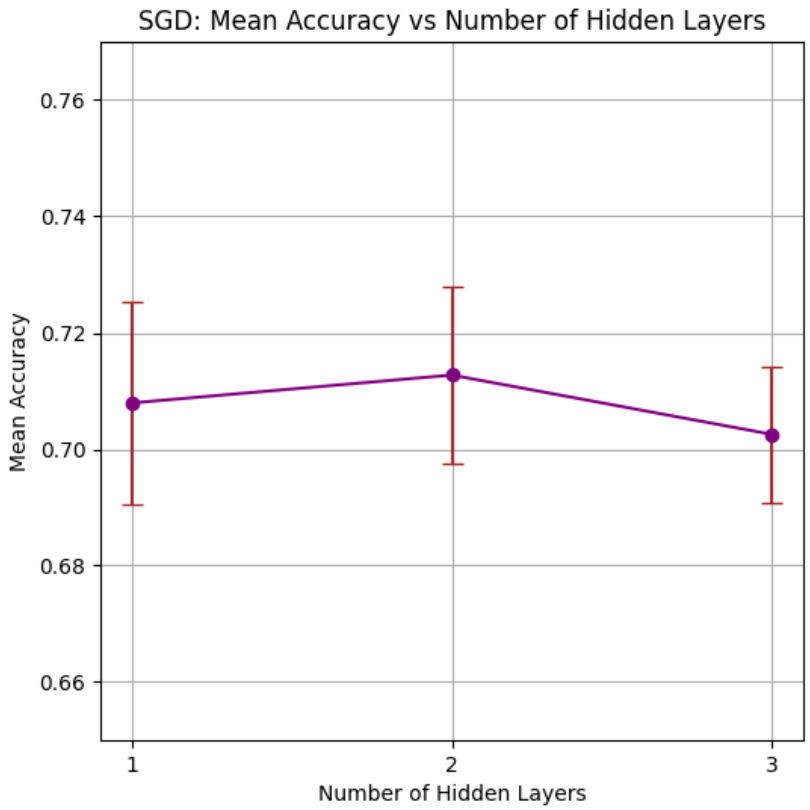


Fig5: Number of Hidden Layers vs Accuracy in FCNN with using 20 neurons in each layer, Learning Rate as 0.01 & SGD optimizer

- 1 hidden layer → Mean Accuracy: 0.70796 → 95% CI: (0.69047, 0.72545)
- 2 hidden layers → Mean Accuracy: 0.71275 → 95% CI: (0.69746, 0.72804)
- 3 hidden layers → Mean Accuracy: 0.70257 → 95% CI: (0.69088, 0.71426)

Best Number of Hidden Layers = 2

5.4.3 Trends in Observation

The experiment exploring the impact of the number of hidden layers on the FCNN's performance, while keeping the number of neurons per layer at 20 and the learning rate at 0.1 with the SGD optimizer, showed a slight variation in mean accuracy across the tested configurations (1, 2, and 3 hidden layers).

The model with one hidden layer achieved a mean accuracy of approximately 0.70796. Increasing the number of hidden layers to two resulted in a slightly higher mean accuracy of around 0.71275. However, further increasing the number of hidden layers to three led to a decrease in the mean accuracy to approximately 0.70257, which was lower than both the one-hidden-layer and two-hidden-layer configurations.

The 95% confidence intervals for all three configurations showed a degree of overlap, suggesting that the differences in mean accuracy might not be statistically significant. The widest confidence interval was observed for the one-hidden-layer configuration, indicating potentially higher variability in its performance compared to the models with two and three hidden layers, which exhibited slightly narrower intervals.

5.4.4 Explanation of Trends

The observed trend suggests that for this specific problem and hyperparameter settings, increasing the depth of the network beyond a certain point does not necessarily lead to improved performance and can even be detrimental.

A **single hidden layer** might have sufficient capacity to learn the underlying relationships in the abalone dataset. While it achieved a reasonable accuracy, the wider confidence interval could indicate that its performance is more sensitive to the specifics of the training data and initialization.

Introducing a **second hidden layer** slightly improved the mean accuracy. This could be because the additional layer allows the network to learn more complex and hierarchical representations of the input features, potentially capturing more nuanced patterns relevant for age classification. The narrower confidence interval also suggests a more stable learning process.

However, adding a **third hidden layer** resulted in a decrease in mean accuracy. This could be attributed to several factors, including the increased risk of overfitting, especially with a relatively small dataset. Deeper networks have more parameters, which can lead to the model memorizing the training data rather than learning generalizable features, especially if not adequately regularized. Additionally, deeper networks can be more challenging to train effectively due to issues like vanishing or exploding gradients, even with a carefully chosen learning rate.

The fact that the confidence intervals overlap suggests that the performance differences between the different numbers of hidden layers might not be statistically significant, implying that for this particular setup, the depth of the network has a relatively limited impact on the overall performance.

5.4.5 Conclusion from Hidden Layers Tuning

Based on the experimental results, a Fully Connected Neural Network with **two hidden layers achieved the highest mean accuracy** for predicting abalone age classes when using 20 neurons per layer, a learning rate of 0.1, and the SGD optimizer. While the improvement over a single hidden layer was marginal, the decrease in performance with three hidden layers suggests that increasing the network depth beyond two layers might lead to overfitting or training instability for this specific task and hyperparameter configuration.

Therefore, considering the slight improvement in mean accuracy and the relatively stable performance indicated by the confidence interval, **two hidden layers appear to be the optimal depth** for the FCNN model under these specific conditions. This finding will guide the subsequent steps in the project, where this network depth can be considered while exploring other optimizers or further refining other hyperparameters. It is important to acknowledge that these conclusions are specific to the tested range of hidden layers and the fixed values of other hyperparameters.

5.5 Step 5: Optimizer Comparison: Hyper Parameter Tuning

Optimizers are crucial in training neural networks, as they influence the speed and quality of convergence during weight updates. To complete the hyperparameter tuning process for our Fully Connected Neural Network (FCNN), we evaluated the impact of different optimization algorithms on model performance.

We replace the SGD optimizer with Adam and repeat all the previous experiments as described below:

5.6 Repeat Step 2: Hidden Layer Neurons for Adam Optimizer

5.6.1 Experiment Setup

Same as step 2 above, only the Optimizer is replaced by Adam (Adaptive Moment Estimation) instead of SGD (Stochastic Gradient Descent) optimizer.

- Model Architecture: A simple feedforward neural network with:
  - One input layer (with features derived from abalone measurements),
  - One hidden layer
  - One output layer with softmax activation for multiclass classification.
- Hyperparameter Tuned: Number of neurons in the hidden layer.
- Repetitions of experiment: 10 runs with different initial weights
- Metric Evaluated: Mean Accuracy on test set with 95% Confidence Intervals.
- Fixed Parameters: Optimizer (Adam), learning rate (0.01), batch size (32) and number of epochs (100) are held constant during this experiment.

5.6.2 Observations

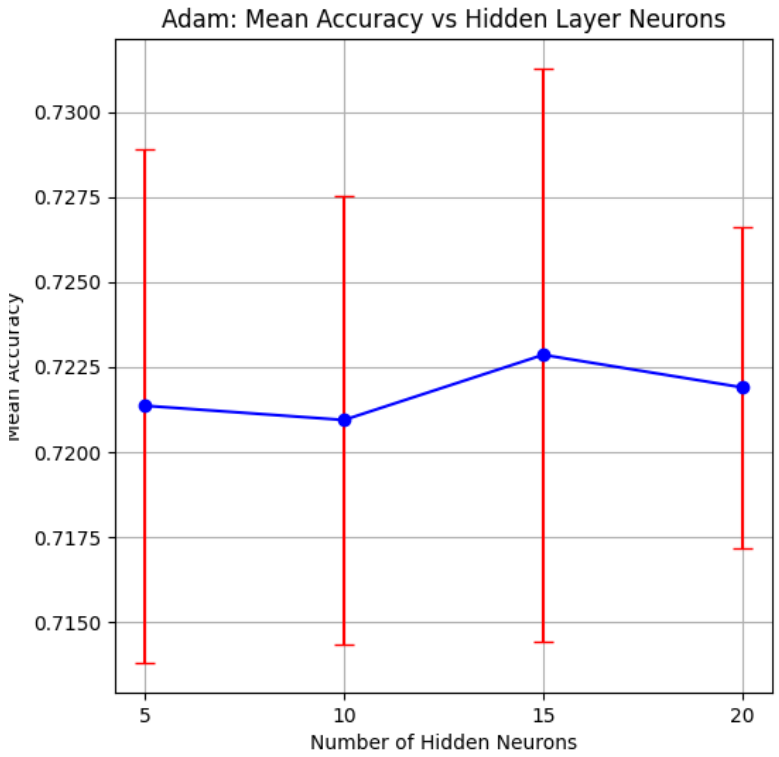


Fig6: Hidden Layer Neurons vs Accuracy in FCNN with 1 hidden layer & Adam optimizer

Using a single hidden layer & Adam optimizer, we varied the number of hidden layer neurons & calculated the resulting Mean Accuracies and 95% Confidence Intervals (CI):

- 5 Neurons → Mean Accuracy: 0.72136 → 95% CI: (0.71380, 0.72892)
- 10 Neurons → Mean Accuracy: 0.72095 → 95% CI: (0.71435, 0.72754)
- 15 Neurons → Mean Accuracy: 0.72286 → 95% CI: (0.71445, 0.73127)
- 20 Neurons → Mean Accuracy: 0.72190 → 95% CI: (0.71717, 0.72663)

Best number of hidden neurons = 15

5.6.3 Trends in Observation

When using the Adam optimizer with a single hidden layer FCNN, varying the number of hidden neurons from 5 to 20 resulted in relatively small fluctuations in the mean accuracy. The mean accuracy started at approximately 0.72136 with 5 neurons, slightly decreased to 0.72095 with 10 neurons, then increased to the highest observed value of 0.72286 with 15 neurons, and finally decreased slightly to 0.72190 with 20 neurons.

The 95% confidence intervals for all tested numbers of hidden neurons were relatively narrow and exhibited a substantial overlap. This suggests that the Adam optimizer led to more stable and consistent performance across different numbers of neurons compared to the SGD optimizer in the earlier experiment. The narrow confidence intervals also imply that the differences in mean accuracy observed across the different neuron counts are likely not statistically significant.

5.6.4 Explanation of Trends

The relatively stable performance observed with the Adam optimizer across different numbers of hidden neurons, compared to the more pronounced trend seen with SGD, can be attributed to the adaptive learning rate nature of Adam. Adam adjusts the learning rate for each weight based on the estimates of the first and second moments of the gradients. This adaptive mechanism often makes the training process less sensitive to the choice of the number of neurons in a single hidden layer within a certain range.

The slight increase in mean accuracy as the number of neurons increased from 5 to 15 might still reflect the enhanced capacity of the network to learn more complex relationships with more parameters. However, the subsequent slight decrease with 20 neurons could indicate that for this specific problem and the fixed learning rate, the model might be starting to overfit slightly or that the added complexity is not necessary and doesn't contribute to better generalization.

The narrower confidence intervals observed with Adam suggest that the optimizer provides a more robust and stable training process, less susceptible to the random initialization of weights and leading to more consistent performance across different runs, regardless of the number of neurons in the single hidden layer. This is a known advantage of adaptive optimizers like Adam, which often exhibit better convergence properties and require less manual tuning of the learning rate compared to traditional optimizers like SGD.

5.6.5 Conclusion from Hidden Layer Neuron Count Tuning

When employing the Adam optimizer with a single hidden layer FCNN, the number of hidden neurons had a less pronounced impact on the mean accuracy compared to when the SGD optimizer was used. The highest mean accuracy of approximately 0.72286 was achieved with **15 hidden neurons**.

While the differences in mean accuracy across the tested neuron counts were small and likely not statistically significant, the configuration with 15 neurons showed a slightly better performance. The overall stable performance and narrow confidence intervals across all neuron counts highlight the robustness of the Adam optimizer for this task.

Therefore, based on this experiment, **15 hidden neurons appear to be the optimal choice** for a single hidden layer FCNN when using the Adam optimizer with a learning rate of 0.01. This result suggests that for subsequent experiments with Adam, we can consider 15 neurons as a potentially good starting point, although further tuning of other hyperparameters might still be necessary to achieve the best possible performance.

5.7 Repeat Step 3: Learning-Rate Tuning for Adam

5.7.1 Experiment Setup

- Architecture: 15 Neurons in 1 hidden layer (best performance) from previous
- Learning Rates Tested: 0.0001, 0.001, 0.01, 0.1
- Evaluation Metric: Mean Validation Accuracy on test set with 95% CI

5.7.2 Observations

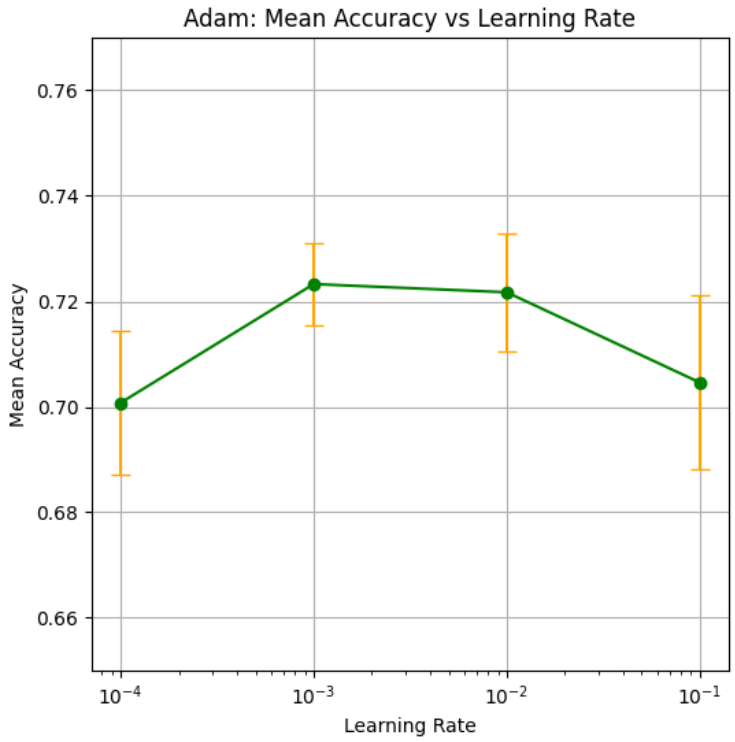


Fig7: Learning Rate vs Accuracy in FCNN with 1 hidden layer using 15 neurons & Adam optimizer

Using 15 neurons, in a single hidden layer & Adam optimizer, we varied Learning Rate and calculated the resulting Mean Accuracies and 95% Confidence Intervals (CI):

- Learning Rate: 0.1 → Mean Accuracy: 0.70461 → 95% CI: (0.68809, 0.72113)
- Learning Rate: 0.01 → Mean Accuracy: 0.72172 → 95% CI: (0.71055, 0.73289)
- Learning Rate: 0.001 → Mean Accuracy: 0.72328 → 95% CI: (0.71548, 0.73108)
- Learning Rate: 0.0001 → Mean Accuracy: 0.70078 → 95% CI: (0.68715, 0.71441)

Best Learning Rate: 0.001

5.7.3 Trends in Observation:

When the learning rate was varied for a single hidden layer FCNN with 15 neurons using the Adam optimizer, the mean accuracy exhibited a non-linear relationship with the learning rate. The lowest mean accuracy was observed at the extreme learning rates: approximately 0.70078 with a learning rate of 0.0001 and around 0.70461 with a learning rate of 0.1.

The mean accuracy improved as the learning rate increased from 0.0001 to 0.001, reaching the highest observed value of approximately 0.72328. Further increasing the learning rate to 0.01 resulted in a slight decrease in mean accuracy to around 0.72172.

The 95% confidence intervals varied in width across the tested learning rates. The widest confidence intervals were associated with the learning rates of 0.1 and 0.0001, suggesting higher variability in performance at these settings. The learning rates of 0.01 and 0.001 had relatively narrower confidence intervals, indicating more stable performance.

5.7.4 Explanation of Trends:

The observed trends highlight the sensitivity of the Adam optimizer's performance to the learning rate, although the optimal range appears to be different from that of SGD.

A **very low learning rate (0.0001)** with Adam, similar to SGD, likely leads to slow convergence, preventing the model from effectively learning the underlying patterns within the fixed number of training epochs.

Increasing the **learning rate to an optimal value (0.001)** allowed the Adam optimizer to effectively update the network weights, leading to the highest mean accuracy and stable performance (narrow confidence interval). Adam's adaptive learning rate mechanism likely benefits from this specific step size, allowing for efficient descent towards a good minimum in the loss landscape.

A **slightly higher learning rate (0.01)** still yielded good performance, but the slight decrease in mean accuracy compared to 0.001 might indicate that the learning rate is becoming too aggressive, potentially causing the optimizer to overshoot the optimal parameter values or oscillate around them.

A **high learning rate (0.1)** resulted in a noticeable drop in mean accuracy and a wider confidence interval, suggesting that the step size is too large, leading to unstable training and hindering the model from converging to a good solution. Adam's adaptive nature couldn't fully compensate for such a large initial learning rate in this scenario.

5.7.5 Conclusion from Learning-Rate Tuning for Adam:

The optimal learning rate for the single hidden layer FCNN with 15 neurons when using the Adam optimizer was found to be **0.001**, which yielded the highest mean accuracy and a relatively narrow confidence interval, indicating stable performance.

This optimal learning rate for Adam (0.001) is notably different from the optimal learning rate found for SGD (0.01) in the earlier experiment. This difference underscores the importance of tuning the learning rate specifically for each optimizer, as their adaptive mechanisms handle the learning process differently. Adam's adaptive learning rate often allows for the use of larger learning rates compared to traditional SGD, but in this specific application, a smaller learning rate within the tested range proved to be most effective.

The results from this experiment confirm that the choice of optimizer significantly impacts the optimal hyperparameter settings. For subsequent experiments using the Adam optimizer, a learning rate of 0.001 will be used as a fixed parameter when exploring other aspects of the model architecture.

5.8 Repeat Step 4: Number of Hidden Layers Tuning for Adam

5.8.1 Experiment Setup

- Architecture: 15 Neurons in 1 hidden layer, Learnin rate 0.001 & Adam optimizer
- Hidden Layers varied: 1, 2 and 3
- Evaluation Metric: Mean Validation Accuracy on test set with 95% CI

5.8.2 Observations

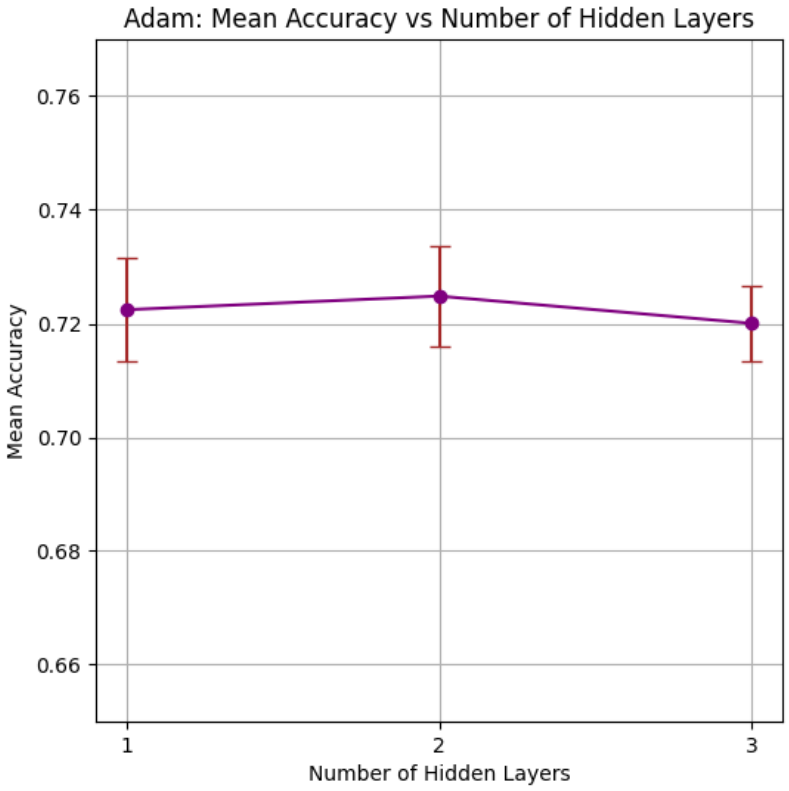


Fig8: Hidden Layes vs Accuracy with 1 hidden layer using 15 neurons, Learning Rate 0.001 & Adam

Using 15 neurons, in a single hidden layer & Adam optimizer, we varied Learning Rate and calculated the resulting Mean Accuracies and 95% Confidence Intervals (CI):

- Hidden Layers: 1 → Mean Accuracy: 0.72244 → 95% CI: (0.71343, 0.73146)
- Hidden Layers: 2 → Mean Accuracy: 0.72484 → 95% CI: (0.71599, 0.73368)
- Hidden Layers: 3 → Mean Accuracy: 0.72005 → 95% CI: (0.71348, 0.72662)

Best Hidden Layers: 2

5.8.3 Trends in Observation

When the number of hidden layers was varied (1, 2, and 3) in the FCNN using 15 neurons per layer and the Adam optimizer with a learning rate of 0.001, the mean accuracy showed a slight increase from one to two hidden layers, and then a decrease with three hidden layers.

The model with one hidden layer achieved a mean accuracy of approximately 0.72244. Increasing the number of hidden layers to two resulted in a slightly higher mean accuracy of around 0.72484. However, further increasing the number of hidden layers to three led to a decrease in the mean accuracy to approximately 0.72005, which was the lowest among the three configurations.

The 95% confidence intervals for all three configurations were relatively narrow and showed substantial overlap, suggesting a consistent level of performance and that the observed differences in mean accuracy might not be statistically significant. The model with two hidden layers exhibited a slightly wider confidence interval compared to the other two.

5.8.4 Explanation of Trends

The observed trend suggests that for the Adam optimizer with the chosen learning rate and number of neurons per layer, a moderate depth of the network (two hidden layers) is slightly beneficial, while going deeper (three hidden layers) might lead to diminishing returns or even a slight degradation in performance.

A single hidden layer provides a baseline level of complexity for learning the relationships in the data.

Introducing a second hidden layer might allow the network to learn more intricate and hierarchical features, leading to a small improvement in the classification accuracy. Adam's adaptive learning rate likely helps in effectively training this slightly deeper network.

However, increasing to three hidden layers appears to slightly decrease the mean accuracy. This could be due to the increased number of parameters making the model more prone to overfitting, even with Adam's adaptive learning rates. Alternatively, the complexity of the relationships in the abalone dataset that can be effectively learned might not necessitate such a deep architecture, and the added layers might introduce noise or make the optimization process slightly more challenging.

The narrow and overlapping confidence intervals suggest that the impact of the number of hidden layers within this range is relatively small when using the Adam optimizer with the tuned learning rate and number of neurons.

5.8.5 Conclusion from Hidden Layers Tuning:

Based on the experimental results using the Adam optimizer with a learning rate of 0.001 and 15 neurons per layer, a Fully Connected Neural Network with two hidden layers achieved the highest mean accuracy for predicting abalone age classes.

While the improvement over a single hidden layer was minimal, the slight decrease in performance with three hidden layers suggests that increasing the network depth beyond two layers is not beneficial for this specific task and hyperparameter configuration when using Adam.

Therefore, considering the slightly higher mean accuracy, two hidden layers appear to be the optimal depth for the FCNN model when using the Adam optimizer with a learning rate of 0.001 and 15 neurons per layer. This configuration will be considered as the best performing architecture identified through our hyperparameter tuning process.

6. Final Model Evaluation

Goal of the Model is to predict **RingAgeClass**, which classifies abalones into 4 age-based categories:

- **Class 0**: Young
- **Class 1**: Young-Adult
- **Class 2**: Adult
- **Class 3**: Old

The final model using SGD optimizer gives a Mean Accuracy of 0.71275 with 95% Confidence Interval between 0.69746 & 0.72804. It uses 20 Neurons in each of its 2 hidden layers and uses a learning rate of 0.01.

While, the final model using Adam optimizer gives a Mean Accuracy of 0.72484 with 95% Confidence Interval between 0.71599 & 0.73368. It uses 15 Neurons in each of its 2 hidden layers and uses a learning rate of 0.001.

6.1 Optimum Model

Therefore, the final model using **Adam optimizer, 15 Neurons in each of its 2 hidden layers and a learning rate of 0.001** is clearly the most optimum multiclass classifier of abalones' ring-based age calculations and classification into 4 corresponding classes.

Further, we are going to plot the following visualizations to illustrate the final model evaluation:

- Confusion Matrix: Shows good classification across all classes with some overlap.
- ROC Curves: Each class demonstrates area under curve values indicating satisfactory separability.

7. Confusion Matrix

Confusion Matrix for Model Evaluation using 15 neurons in each of 2 layers hidden, Learning Rate as 0.001 & Adam optimizer is drawn below:

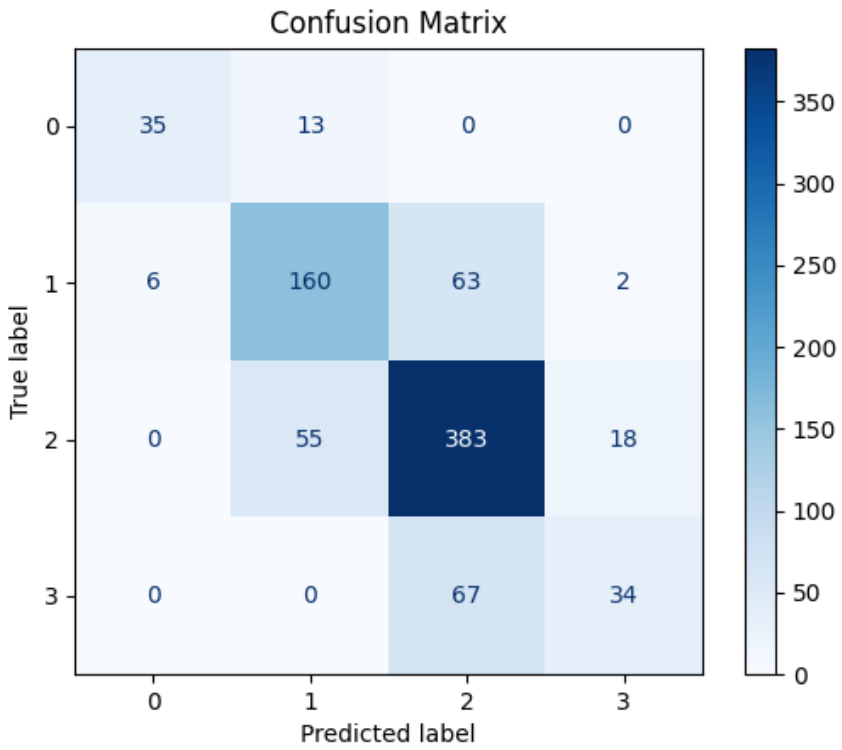


Fig9: Confusion Matrix for Model Evaluation using 15 neurons in each of 2 layers hidden, Learning Rate as 0.001 & Adam optimizer.

NOTE:

- **Diagonal values (bold)** represent **correct predictions**.
- **Off-diagonal** values are **misclassifications**.

7.1 Class-by-Class Interpretation

Class 0 (Young Abalones)

- **Correctly Predicted:** 35
- **Misclassified as Class 1:** 13
- **Observation:** Small sample size but relatively good performance. Some confusion with Class 1. This makes sense biologically if physical traits are similar in early growth stages.

Class 1 (Young-Adult)

- **Correctly Predicted:** 160
- **Misclassified as Class 0:** 8

- **Misclassified as Class 2:** 63
- **Misclassified as Class 3:** 2
- **Observation:** Most confused with **Class 2**, possibly due to overlapping features like weight and shell size as they mature.

Class 2 (Adult)

- **Correctly Predicted:** 383
- **Misclassified as Class 1:** 55
- **Misclassified as Class 3:** 18
- **Observation:** Strong performance. This class has the **highest correct count**, meaning the model finds adults easier to distinguish—likely due to distinct patterns in physical features.

Class 3 (Old)

- **Correctly Predicted:** 34
- **Misclassified as Class 2:** 67
- **Observation:** Heavily misclassified as Class 2. Indicates **difficulty in separating older abalones** from adults—perhaps due to diminishing feature contrast as abalones age.

7.2 Key Takeaways

- **Good performance on Class 2**, the most populated and distinct group.
- **Overlap between neighboring classes**, especially Class 1 and Class 2, and between Class 2 and Class 3.
- **Biological Reasoning:** Adjacent age classes often share similar physical attributes, leading to confusion.
- **Model Strength:** Learns well from features like weight and shell thickness.
- **Model Limitation:** Might benefit from better feature engineering or class rebalancing.

8. ROC Curves

The **Receiver Operating Characteristic (ROC) curve** is used to visualize the **trade-off between the true positive rate (sensitivity)** and **false positive rate (1 - specificity)** for different classification thresholds. For **multiclass classification**, the ROC is typically plotted using a **One-vs-Rest (OvR)** approach.

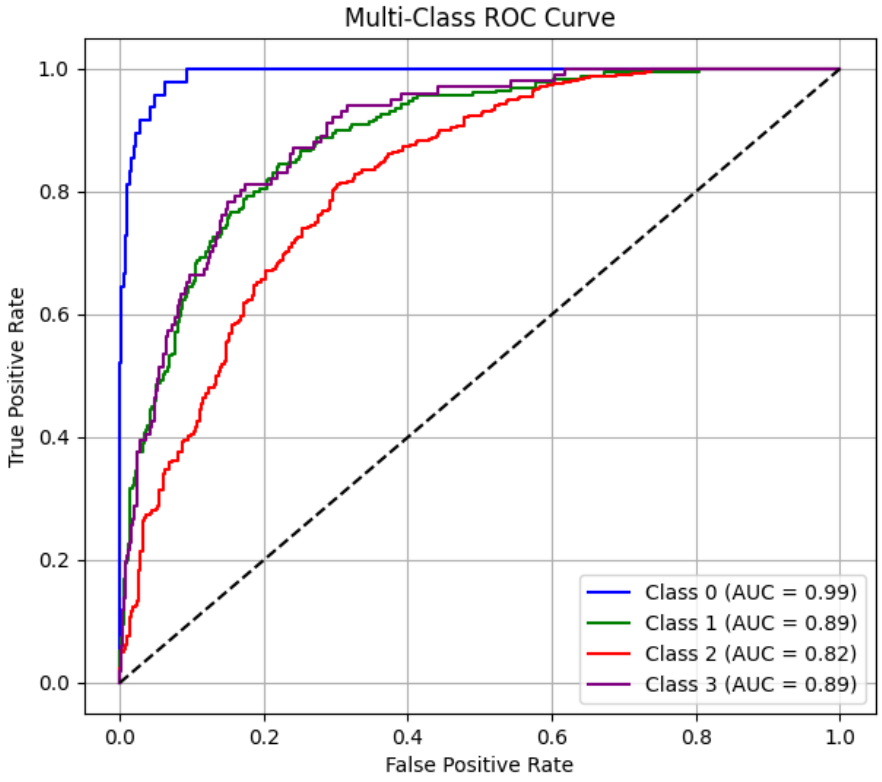


Fig10: Multi Class ROC Curve

Class	Age Group	AUC Score	Interpretation
0	Young	**0.99**	Outstanding separability
1	Young-Adult	**0.89**	Strong performance
2	Adult	**0.82**	Good but could improve
3	Old	**0.89**	Strong, though some confusion remains

8.1 Class-by-Class ROC Analysis

Class 0 (Young) – AUC = 0.99

- Excellent classification capability.

- Curve is closest to the top-left corner, indicating near-perfect prediction with very few false positives or false negatives.
- Likely due to distinct features (e.g., size and weight) in the youngest abalones.

Class 1 (Young-Adult) – AUC = 0.89

- Very good performance.
- Some overlap with Class 2 is expected due to transitional features.
- Model captures subtle changes in physical measurements fairly well.

Class 2 (Adult) – AUC = 0.82

- The **lowest AUC among all classes**, but still solid.
- Reflects what we saw in the **confusion matrix**, where many samples from other classes were misclassified as Class 2.
- Indicates that adult abalones share overlapping features with both younger and older classes, making boundaries blurrier.

Class 3 (Old) – AUC = 0.89

- Good predictive capability.
- Some confusion with Class 2, which is biologically plausible, as physical growth changes may slow or stabilize with age.

8.2 What the ROC Curve Confirms

- The model performs **especially well at identifying youngest class (Class 0)**.
- Class 1 and Class 3 are also well-separated, though with minor overlap.
- Class 2, while the most populated and best classified in the confusion matrix, shows **more overlap in terms of ROC**, suggesting feature similarity with surrounding age groups.

8.3 Summary Insights

- **ROC Curves complement the confusion matrix**, giving a threshold-independent view of how well the model distinguishes each class.
- **High AUC values (all above 0.80)** across the board demonstrate a **robust classifier**, especially in the context of real-world biological data where exact age boundaries are naturally fuzzy.
- Model's decision boundaries are **strong for Class 0** and **Class 1**, a bit **softer for Class 2**, which makes sense considering age group overlaps.

9. Results and Discussion

This project successfully demonstrates how a well-tuned Fully Connected Neural Network can predict abalone age categories based on physical measurements.

9.1 Results:

The results highlight the critical importance of hyperparameter tuning. The Adam optimizer outperformed SGD. The number of hidden neurons that positively impacted performance is 15. The learning-rate of 0.001 proved ideal for convergence. Making hidden layers to 2 improved performance.

Our best-performing model using Adam optimizer, 2 hidden layers with 15 neurons each, and a learning rate of 0.001, achieving a final Mean Test Accuracy of 0.72484 with 95% Confidence Interval between 0.71599 & 0.73368.

9.2 Conclusion:

The confusion matrix and ROC curves provide evidence that the model can effectively distinguish between the four age classes, although minor misclassifications remain, especially between adjacent classes.

9.3 Further Discussion:

Limitations include the use of a fixed number of epochs and batch size, which could be further tuned. Moreover, advanced regularization techniques or dropout could be explored to prevent overfitting.

Also, future work could involve ensemble techniques, CNNs for more complex feature extraction, and incorporating environmental variables to enhance model robustness.

10. References & Source Overview

This study leverages a well-rounded collection of academic, data-centric, and software-based sources that collectively inform both the theoretical foundations and the implementation details of the project.

10.1 Dataset Source

10.1.1 Dua, D. & Graff, C. (2019):

UCI Machine Learning Repository: Abalone Dataset

Retrieved from http://archive.ics.uci.edu/ml.

Irvine, CA: University of California, School of Information and Computer Science.

This dataset serves as the cornerstone for the project. It provides biometric measurements of abalones, including weight, shell size, and sex, which are used as features in the neural network classification task. The categorization into ring-age classes is directly derived from this source.

10.2 Deep Learning Frameworks

10.2.1 Chollet, F. (2015):

Keras

Retrieved from https://keras.io

Keras, a high-level neural networks API, was used to build and train the Fully Connected Neural Networks (FCNNs). Its modular nature facilitated experimentation with different architectures, optimizers, and learning rates.

10.2.2 Abadi, M., et al. (2016).

TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems

Retrieved from https://www.tensorflow.org/

TensorFlow forms the backend engine that powers the execution of the models built with Keras. It offers efficient computation graphs and hardware acceleration, which proved beneficial for the large number of training iterations.

10.3 Theoretical Foundations

Goodfellow, I., Bengio, Y., & Courville, A. (2016)

Deep Learning. MIT Press.

This book provides the theoretical framework for understanding FCNNs, activation functions, backpropagation, and optimization strategies such as SGD and Adam. Hyperparameter tuning approaches referenced throughout the project (e.g., adjusting learning rates, hidden layers, and neuron counts) are rooted in concepts detailed in this text.

10.4 Evaluation Metrics and Methodology

The application of **Confusion Matrices**, **ROC Curves**, and **AUC Scores** as evaluation tools is grounded in conventional machine learning performance assessment practices. These are considered best practices in multi-class classification, and their use in this project is consistent with academic standards found in major ML publications.

10.5 Appendix

- All plots saved in the directory: /home/plots/
- Codebase and data preprocessing scripts are maintained for reproducibility in GitHub: https://github.com/anodiamadmin/AIML
- Codebase and data preprocessing scripts are maintained for reproducibility in Google Colab: https://colab.research.google.com/drive/1MB2ExeNSmVIAD4bRwlIxCbeeIzhIYzKD