Analysis of Machine Learning Methods for Predicting Abalone Age from Physical Measurements

Minjun Kim
z5477749@ad.unsw.edu.au

Zheyu He
z5487658@ad.unsw.edu.au

***Abstract*—The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope - a boring and destructive task. However, by leveraging machine learning methods, we aim to predict abalone age using easily obtainable physical measurements, such as length, diameter, and various weight attributes. We begin with data exploration: visualising the dataset's class and feature distributions, and projecting the data onto two PCA principal components. We then develop a range of machine learning models for classification purposes, including pre/post-pruned Decision Trees, ensemble methods including Random Forest, Gradient Boosting and XGBoost, alongside simple neural networks trained with Adam and SGD. However, two questions remain: 1. Which model is the most suitable for our dataset and goal? 2. How can we determine the hyper-parameters that optimise each model's predictive performance? To address this, we evaluate model performance across different hyper-parameters, such as tree depth and pre/post-pruning strength for Decision Trees, number of trees for Random Forests, learning rate for boosting models, and regularisation/dropout rate for neural networks. Model performance is benchmarked against metrics such as AUC/ROC and F1-score. Overall, this study investigates the effects of hyper-parameter optimisation and dimensionality reduction on model performance, allowing us to identify the most effective approach for accurate prediction of abalone age.**

***Index Terms*—data exploration, trees, ensemble methods, neural networks, hyper-parameter, dimensionality reduction**

I. Introduction

The objective of supervised machine learning is to apply statistical methods to optimise models that can predict class labels based on input features [1]. Decision trees rose in popularity in the 1980s with the development of algorithms like ID3 and CART, providing a classifier expressed as a recursive partition of the instance space [2]. Random forest extends decision trees by creating an ensemble of bootstrapped trees, reducing correlation in feature data to improve predictive performance [3]. Gradient Boosting, in contrast, builds trees sequentially, ultimately designed to improve or reinforce a single weak model by combining it with many other weak models to create a strong model together; XGBoost strengthens poor learning mechanisms iteratively [4]. A newer method, neural networks, have shown very promising results by creating connections between processing elements, where the organisation and weights of the connections determine the output [5].

Despite their potential, neural neural networks face many challenges to enable their full and, at scale, realisation. During training, neural networks are prone to overfitting, particularly when trained on training data contemplating small intra-class variation, or excessive training using a small training set [6]. Neural networks also often require extensive and representative datasets for effective generalisation, which is not always practical in real-world applications [7]. Furthermore, they were very sensitive to hyperparameter choices, including learning rates, depth and regularisation strength. As a result, good performance requires careful tuning - a complicated and time-consuming procedure that includes picking the best algorithm and creating the best model architecture, despite the output targets and input characteristics [8]. In applied settings, neural networks suffer from four classical challenges: systematic generalisation, catastrophic forgetting, few-shot learning and multi-step reasoning [9].

Beyond challenges plaguing neural networks, several broader issues arose in applying machine learning methods to the biological task of predicting abalone age. A central challenge involved managing the trade-off between predictive precision and the ability of models to generalise [10]. The abalone dataset also posed challenges prevalent in biological datasets, including imbalanced class distributions, overlap between classes, measurement noise, and feature correlation, all of which complicate the learning problem [11]. Ensemble methods attempt to partially alleviate these issues through increased model flexibility and reduced variance, though at the expense of increased computational cost, uninterpretability of results, and the fact that the models themselves cannot be explained [12]. Similarly, Boosting models were prone to overfitting when hyperparameters like tree depth, learning rates and training iterations were not controlled properly [13].

The prediction of abalone age group from physical measurements is a relevant scientific problem, yet it remains largely unexplored by modern machine learning methods. The dataset's complexity combined with the difficulty of model selection and hyperparameter tuning, renders the problem particularly challenging. This motivates a deep investigation of the different modelling approaches, including how preprocessing and hyperparameter optimisation change the nature of the task. In addition, to evaluate whether such insights generalise beyond biological datasets, we extend our analysis to the Contraceptive Method Choice dataset. As students, we will take on this challenge to both identify the most effective

models across two classification tasks, and to strengthen our understanding of how different models and hyperparameter choices compare in predictive performance.

In this project, we investigate the problem of predicting abalone age group through a rigorous evaluation of a range of machine learning methods, including pre/post-pruned Decision Trees, ensemble methods including Random Forest, Gradient Boosting and XGBoost, alongside simple neural networks trained with Adam and SGD. The dataset's target values are transformed into a four-class classification problem, and model performance is evaluated using metrics such as accuracy score, F1-score and ROC/AUC. We study the influence of key hyperparameters including tree depth, number of trees learning rate, and neural network regularisation. We also examine the impact of pre/post-pruning on decision trees, and dimensionality reduction through PCA. For the CMC dataset, we provide the relevant data visualisation and apply the two best models: XGBoost and neural network, evaluating their predictive performance using similar metrics, Ultimately, we aim to provide a uniform comparison of different modelling approaches to identify the most effective models across both datasets.

The rest of the paper is organised as follows. Section 2 outlines the methodology, including a description of the abalone dataset, the data preprocessing steps, overview of the methods considered, workflow diagram, software suites, and experiment setting. Section 3 lays out the results, reporting the performance of different models across hyperparameter configurations as well as the effects of pre/post-pruning, PCA dimensionality reduction and neural network regularisation. Section 4 summarises our findings through a discussion, noting the implications of the results and limitations of our methodology. Section 6 concludes the study with the main contributions to the field and specifies directions for potential future research.

II. Methodology

A. Data

Our investigation uses the UCI Abalone dataset [14]. The tabular dataset contains 4417 instances with no missing values, with 8 features including *Sex, Length, Diameter, Height, Whole weight, Shucked weight, Viscera weight*, and *Shell weight*. *Sex* is a categorical feature with three levels: M, F, I. All other features are continuous. The target value is *Rings* (continuous, +1.5 gives the age in years).

We also explore the UCI Contraceptive Method Choice dataset [15]. This dataset contains 1473 instances with no missing values, with 8 features including *wife age, wife education, husband education, number of children, wife's religion, wife's working status, husband's occupation, standard of living index* and *media exposure*, with most features being nominally encoded, categorical features. The target value is *contraceptive method*.

For both datasets, preprocessing was minimal, as decision trees and related ensemble methods do not require many preprocessing steps, nor feature selection. Hence, for the most part, the models were trained on unnormalised datasets with all features, split to a 80-20 train-test split. The only exceptions were with training neural networks, where we wished to compare the effects of feature selection and normalisation, and performing PCA-decomposition, where the *MinMaxScaler* was fitted on the features before applying PCA.

B. Overview of the Methods

In this work, we compared a set of classical tree-based algorithms—Decision Trees, Random Forests, Gradient Boosting Machines, and XGBoost—with fully connected Neural Networks trained using stochastic gradient descent (SGD) and the Adam optimiser. Each method offers different modelling assumptions, optimisation strategies, and theoretical properties, allowing a comprehensive evaluation across both interpretable and high-capacity models.

1) Decision Tree Classifier: Decision tree classifiers are intuitive supervised learning models that partition the feature space into axis-aligned regions. A tree consists of internal decision nodes and terminal leaf nodes. At each decision node, the model selects a feature and threshold that split the dataset into two subsets. The quality of a split is measured by the reduction in impurity, where impurity quantifies the heterogeneity of class labels.

For a subset S with class proportions p_i, common impurity measures include the entropy

$$Entropy(S) = -\sum_i p_i \log_2 p_i, \tag{1}$$

and the Gini index

$$Gini(S) = 1 - \sum_i p_i^2. \tag{2}$$

For a candidate split producing subsets S_1 and S_2, the impurity reduction (also called information gain) is defined as

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{3}$$

where $Entropy(S)$ can be easily replaced by $Gini(S)$. The algorithm recursively chooses the split with the largest information gain until a stopping criterion is met (e.g., minimum leaf size or maximum depth). The resulting tree forms a piecewise-constant approximation of the underlying decision boundary.

2) Random Forest Classifier: Random Forests extend decision trees by constructing an ensemble of decision trees, each trained on a bootstrap sample S drawn with replacement. To further decorrelate the trees, the algorithm alternates node splitting criteria, and selects a random subset of features $\mathcal{F}_m$ (with $|\mathcal{F}_m| = m$) at each split, and maximises impurity reduction only over $\mathcal{F}_m$.

Formally, the Random Forest classifier prediction is given by

$$\hat{y} = \text{mode}\left(h_1(x), \ldots, h_B(x)\right), \tag{4}$$

where each h_b is a decision tree. In simple terms, it is simply a majority vote between all the decision trees that have been trained. By doing so, Random Forests generally reduces variance by averaging many high-variance trees.

3) Gradient Boosting: Gradient Boosting constructs an additive model of the form

$$F_M(x) = \sum_{m=1}^{M} \gamma_m h_m(x), \tag{5}$$

where each base learner $h_m(x)$ is typically a shallow decision tree, and γ_m is a step size. At iteration m, the model fits a new tree to the negative gradient of the loss function $L(y, F_{m-1}(x))$ with respect to the predictions:

$$r_i^{(m)} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F=F_{m-1}}. \tag{6}$$

The algorithm then fits $h_m(x)$ to the pseudo-residuals $r_i^{(m)}$ and computes the optimal scaling factor

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)). \tag{7}$$

This procedure performs functional gradient descent in the space of functions. Learning rate (shrinkage) and tree depth serve as regularisation, controlling the complexity and preventing overfitting.

4) XGBoost: XGBoost improves on classical gradient boosting by incorporating second-order optimisation, explicit regularisation, and efficient tree construction. The objective function at iteration m is approximated by a second-order Taylor expansion:

$$\mathcal{L}^{(m)} \approx \sum_{i=1}^{n} \left[g_i h_m(x_i) + \frac{1}{2} h_i h_m(x_i)^2 \right] + \Omega(h_m), \tag{8}$$

where g_i and h_i are the first and second derivatives of the loss with respect to predictions, and

$$\Omega(h_m) = \lambda \sum_{j=1}^{T} w_j^2 + \alpha \sum_{j=1}^{T} |w_j| \tag{9}$$

is a regularisation term on the leaf weights w_j (with T leaves). For a given tree structure with leaf sets I_j, XGBoost computes the optimal leaf weight in closed form:

$$w_j^\star = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}. \tag{10}$$

The corresponding loss reduction, used for split selection, is

$$\text{Gain} = \frac{1}{2}\left[\frac{\left(\sum_{i \in I_L} g_i\right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i\right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i\right)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma, \tag{11}$$

where γ penalises the creation of new leaves. These improvements allow XGBoost to achieve high accuracy while controlling overfitting and maintaining computational efficiency.

5) Neural Network Classifier: We trained fully connected neural networks, which modelled a function

$$f(x; \Theta) = W_L \sigma(W_{L-1} \sigma(\cdots \sigma(W_1 x + b_1) \cdots) + b_{L-1}) + b_L,$$

where $\sigma(\cdot)$ denotes a nonlinear activation function and Θ collects all learnable parameters. Training minimised a loss function $L(y, f(x; \Theta))$ via iterative optimisation.

a) Stochastic Gradient Descent (SGD): SGD updated parameters using gradients computed on mini-batches:

$$\Theta_{t+1} = \Theta_t - \eta \nabla_\Theta L_t, \tag{12}$$

where η is the learning rate. Although simple, SGD may converge slowly or become trapped in regions of small gradient, particularly on ill-conditioned loss surfaces.

b) Adam Optimiser: Adam improved upon SGD by maintaining first- and second-moment estimates of the gradients:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{13}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2. \tag{14}$$

After bias correction,

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}, \tag{15}$$

parameters were updated as

$$\Theta_{t+1} = \Theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}. \tag{16}$$

These adaptive per-parameter learning rates enabled faster and more stable convergence across a variety of neural architectures and datasets.

C. Framework and Workflow Design

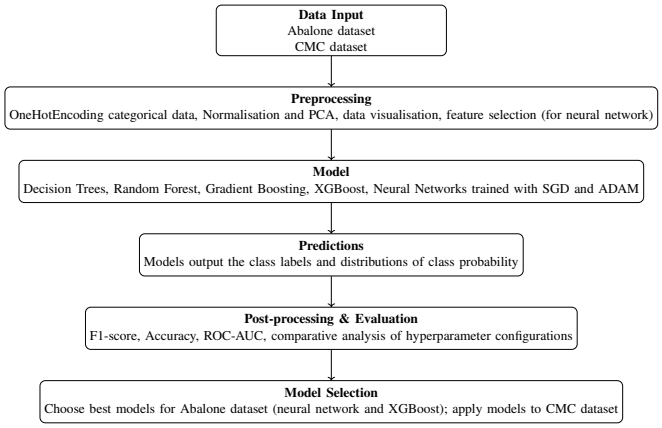

Fig. 1. Workflow diagram for the Abalone and CMC datasets

D. Software Suite

The implementation uses Python as the primary programming language with the following libraries:

Data Processing:

- `pandas`: data manipulation and CSV file handling
- `NumPy`: numerical computations and array operations

Machine Learning Models:

- `scikit-learn`: Decision Trees, Random Forests, Gradient Boosting, MLP Neural Networks, and preprocessing utilities including `OneHotEncoder`, `MinMaxScaler`, and PCA
- `XGBoost`: gradient boosting implementation optimised for performance

Hyperparameter Optimisation:

- `Optuna`: Bayesian hyperparameter optimisation with TPE sampler

Evaluation Metrics:

- `scikit-learn.metrics`: F1 score (micro-averaged) and ROC-AUC score
- `scikit-learn.model_selection`: cross-validation and train-test splitting

Visualisation and Output:

- `matplotlib`: plotting and figure generation
- `pandas`: Excel file export for results tables

E. Experiment Setting

Data Splitting: We use an 80/20 train-test split with stratified sampling to maintain class distribution across splits. To ensure robust evaluation, we conduct experiments across 50 different random splits (random_state 0–49) and report mean and standard deviation of metrics.

Hyperparameter Tuning: We perform hyperparameter optimisation on a single split (random_state=0) using Optuna with the following configuration:

- Sampler: TPE (Tree-structured Parzen Estimator) with 15 startup trials and multivariate sampling enabled
- Pruner: Median pruner with 1 warmup step
- Number of trials: 50–100 depending on model complexity (50 for Decision Trees, 60 for Random Forests and Gradient Boosting, 100 for XGBoost)
- Objective: 5-fold cross-validation F1-micro score
- Parallelisation: enabled (`n_jobs=-1`)

The optimised hyperparameters are then applied across all 50 evaluation splits to assess generalisation performance.

Hyperparameter Search Spaces:

TABLE I
DECISION TREE HYPERPARAMETER SEARCH SPACES

Parameter	Full Tuning	Pre-pruning	Post-pruning
max_depth	[1, 20]	[1, 20]	–
min_samples_leaf	[1, 20]	[1, 20]	–
min_samples_split	[2, 50]	[2, 50]	–
max_leaf_nodes	[10, 300]	[10, 300]	–
ccp_alpha	[0.0, 0.05]	–	[0.0, 0.05]
min_impurity_decrease	–	[0.0, 0.2]	–
criterion	{gini, entropy}	{gini, entropy}	{gini, entropy}

TABLE II
RANDOM FOREST HYPERPARAMETER SEARCH SPACE

Parameter	Range
n_estimators	[50, 500]
max_depth	[3, 20]
min_samples_split	[2, 20]
min_samples_leaf	[1, 10]
criterion	gini, entropy
bootstrap	True, False

TABLE III
GRADIENT BOOSTING HYPERPARAMETER SEARCH SPACE

Parameter	Range
n_estimators	[50, 400]
learning_rate	[0.01, 0.2]
max_depth	[2, 6]
min_samples_split	[2, 20]
min_samples_leaf	[1, 10]
subsample	[0.7, 1.0]

TABLE IV
XGBOOST HYPERPARAMETER SEARCH SPACE

Parameter	Range
n_estimators	[50, 400]
learning_rate	[0.01, 0.3]
max_depth	[2, 6]
min_child_weight	[1, 10]
subsample	[0.6, 1.0]
colsample_bytree	[0.6, 1.0]
gamma	[0, 0.3]
reg_lambda	[0, 1.0]

Neural Network Settings:

- Architecture: default scikit-learn MLP configuration (single hidden layer with 100 neurons)
- Activation function: ReLU
- Solvers evaluated: Adam, SGD
- L2 regularisation (α values: 0.0001, 0.001, 0.01)
- Maximum iterations: 200 (default)

PCA Settings:

- Preprocessing: MinMax scaling applied before PCA
- Variance retention thresholds: 95%, 98%

Evaluation Metrics:

- F1 Score (micro-averaged): primary metric for model comparison, suitable for multi-class imbalanced datasets
- ROC-AUC Score: secondary metric using one-vs-rest strategy for multi-class classification

Label Encoding: For XGBoost and Neural Network experiments, class labels are shifted from 1,2,3,4 to 0,1,2,3 to comply with zero-indexed class requirements.

Reproducibility: All experiments use fixed random states for reproducibility. Model implementations follow an object-oriented design pattern with an abstract base class defining common training, prediction, and evaluation interfaces.

III. RESULTS

In this section, we present the results of our experimental evaluation on the Abalone multi-class classification task. All

experiments are conducted across 50 different train-test splits to ensure robust evaluation, and we report mean and standard deviation for each metric.

A. Exploratory Data Analysis (EDA)

Figure 2 shows the pearson correlation heatmap for the *Abalone* dataset, while figure 3 shows a similar heatmap but for the *CMC* dataset using the Spearman coefficient, due to the presence of mixed nominal/ordinal encoded data.

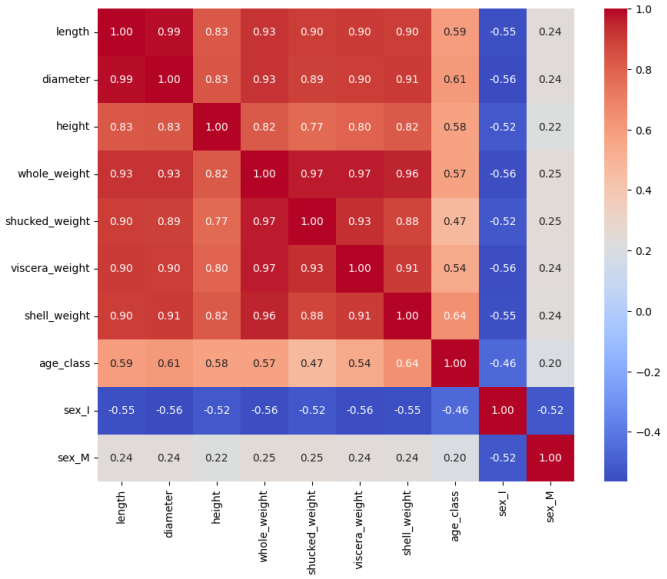

Fig. 2. Pearson correlation heatmap for the Abalone dataset

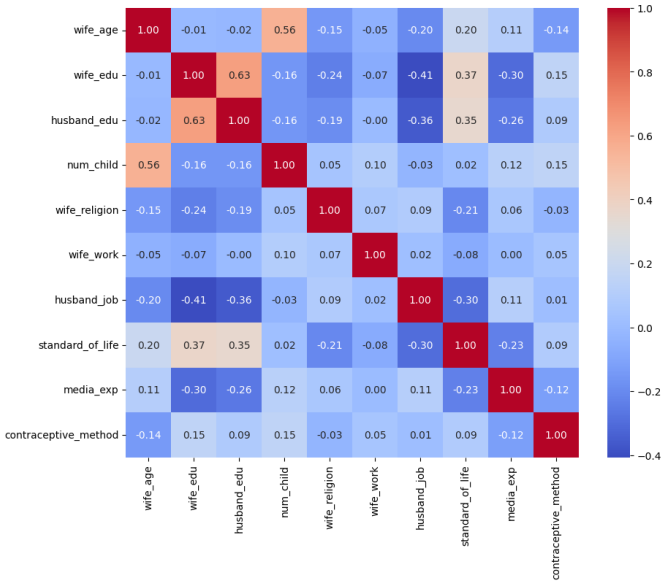

Fig. 3. Spearman correlation heatmap for the CMC dataset

From our heatmaps, we see that for the *Abalone* dataset, *diameter* and *length* were the top 2 features with the highest correlation with our target. Similarly, *wife_edu* and and *num_child* had the highest correlation for the *CMC* dataaset. However, it is important to note that the *CMC* features have a much lower correlation score compared to the *Abalone* features.

For these selected features of the *abalone* dataset, we look into their distribution:

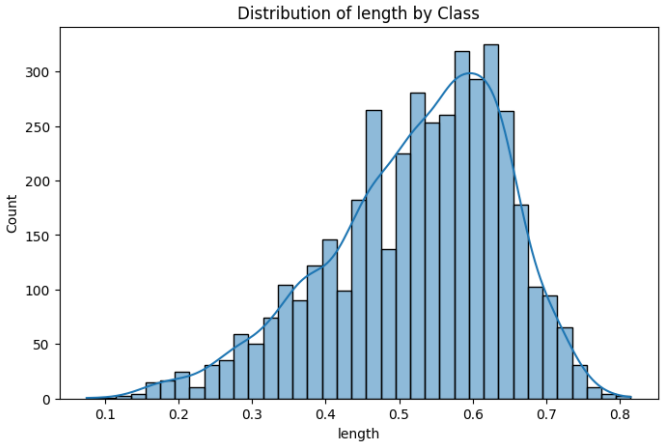

Fig. 4. Distribution of feature *length*

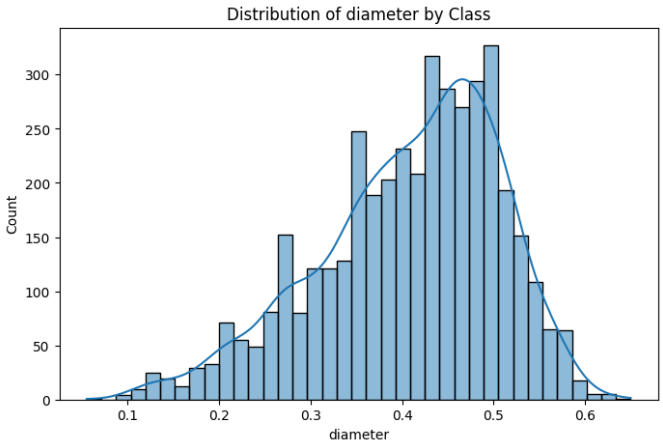

Fig. 5. Distribution of feature *diameter*

Length: The distribution of *length* exhibited a slight negative skew, with values ranging from approximately 0.1 to 0.8. The modal value occurred around 0.55–0.60, with a peak count of approximately 325 observations. The kernel density estimate overlay confirms the near-normal distribution with a subtle left tail.

Diameter: The *diameter* distribution displayed a similar pattern to length, with values ranging from approximately 0.05 to 0.65. The distribution peaked around 0.45–0.50 with approximately 325 observations at the mode. A slight negative skew was observed, with the kernel density estimate revealing a minor secondary concentration around 0.25–0.35.

Similarly for the *CMC* dataset,

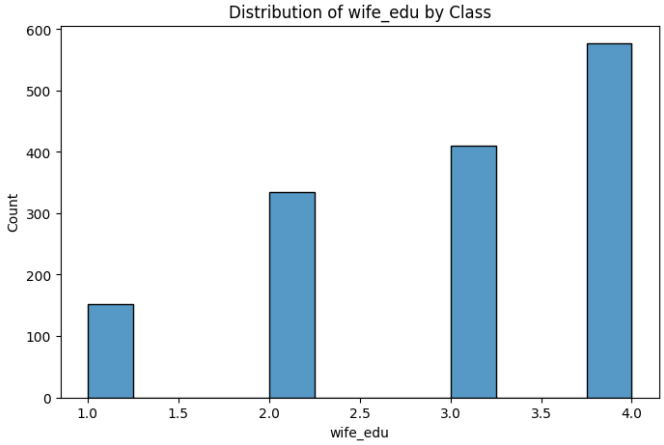

Fig. 6. Distribution of feature *wife_edu*

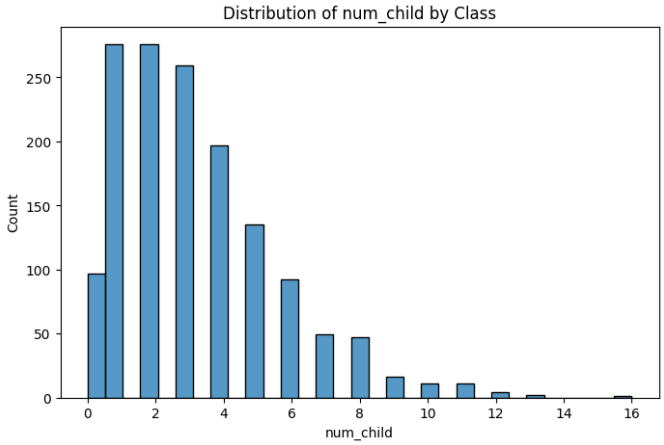

Fig. 7. Distribution of feature *num_child*

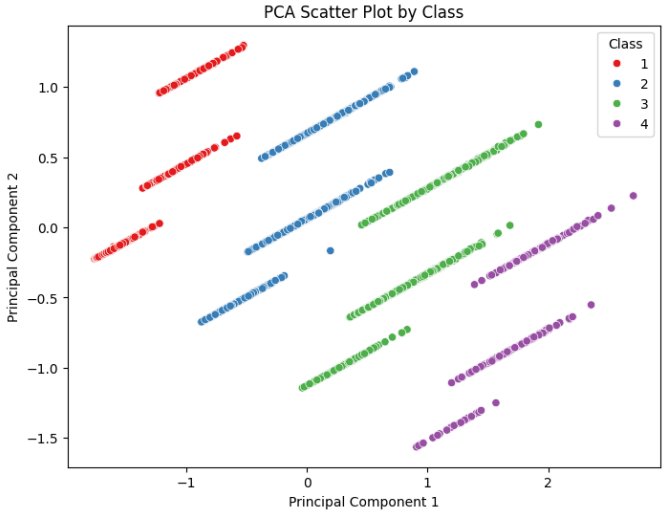

Fig. 8. Scatterplot of PCA features

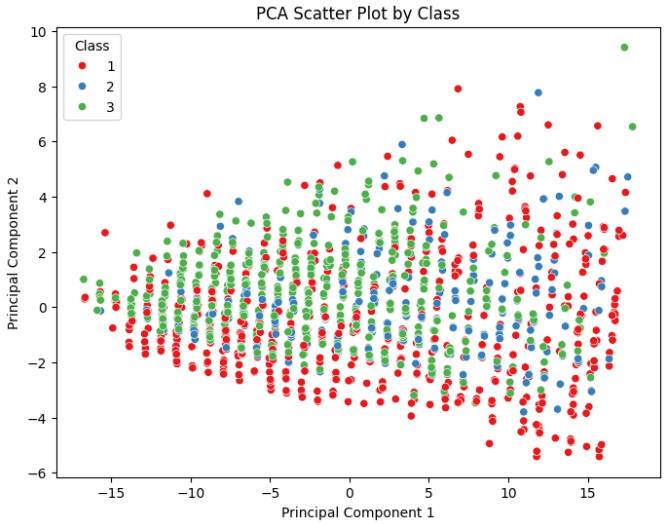

Fig. 9. Scatterplot of PCA features

Wife's Education: The distribution of wife's education level revealed a positive skew toward higher education categories. The highest education level (4) was the most frequently observed, comprising approximately 575 observations, while the lowest education level (1) contained only approximately 150 observations.

Number of Children Ever Born: The distribution of children exhibited a right-skewed pattern characteristic of fertility data. The modal values occurred at 1–2 children, with approximately 275 observations each. Frequency decreased progressively for higher values, with a long tail extending to a maximum of 16 children. The majority of observations fell within the 1–4 children range, while families with 10 or more children represented a small minority of cases.

For both the *Abalone* and *CMC* dataset, these feature distribution analysis is not too much of a concern to us, as trees, ensemble methods, and neural networks tends to be robust to skewness and multicollinearity.

Next, we analyse the distribution of PCA components of each dataset.

Abalone Dataset: The PCA projection revealed a distinctive banded structure, with observations arranged in parallel diagonal stripes across the principal component space. This pattern is characteristic of datasets dominated by discrete or ordinal features, where integer-valued variables create regular spacing in the projected space. Notably, the four classes exhibited clear separation along Principal Component 1, with Class 1 (red) concentrated at lower PC1 values, progressing through Classes 2 and 3, to Class 4 (purple) at higher PC1 values. This ordinal separation suggests that PC1 captures variance strongly associated with the class label, indicating favourable conditions for classification.

CMC Dataset: In contrast, the PCA projection for the contraceptive method dataset revealed substantial overlap among all three classes. The scatter plot displayed a single continuous distribution with no discernible clustering or class separation. Classes 1 (No-use), 2 (Long-term), and 3 (Short-

term) were intermixed throughout the principal component space, suggesting that the first two principal components do not capture features that discriminate between contraceptive method choices. This overlap foreshadows potential difficulty in achieving high classification accuracy and suggests that the relationship between features and contraceptive choice may be complex and non-linear.

Due to this, models trained on datasets with PCA were considered and tested for in the *Abalone* dataset, however, was not tested for in the *CMC* dataset.

B. Model Comparison

Table V presents the performance comparison of Decision Tree, Random Forest, Gradient Boosting, and XGBoost classifiers with Optuna-optimised hyperparameters.

TABLE V
PERFORMANCE COMPARISON OF TREE-BASED MODELS (50 SPLITS)

Model	Train F1	Test F1	Train AUC	Test AUC
Decision Tree	0.640 ± 0.007	0.622 ± 0.013	0.843 ± 0.004	0.823 ± 0.010
Random Forest	0.836 ± 0.004	0.644 ± 0.013	0.975 ± 0.001	0.852 ± 0.007
Gradient Boosting	0.750 ± 0.004	0.645 ± 0.016	0.925 ± 0.001	0.851 ± 0.008
XGBoost	0.699 ± 0.004	**0.647 ± 0.014**	0.901 ± 0.001	0.851 ± 0.008

XGBoost achieves the highest test F1 score of 0.647, followed closely by Gradient Boosting at 0.646 and Random Forest at 0.643. The single Decision Tree underperforms all ensemble methods with a test F1 of 0.622. We observe that ensemble methods improve test F1 by approximately 2-3% over the single Decision Tree. The gap between training and test performance is largest for XGBoost (0.075) and smallest for Decision Tree (0.019), indicating that ensemble methods fit the training data more aggressively but still generalise better overall. All models exhibit low standard deviations across the 50 splits, indicating stable and reproducible results.

A few visualisations of the best performing decision tree are also given below:

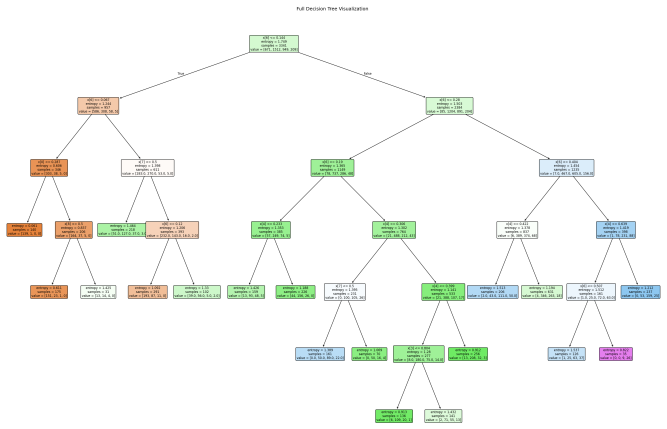

Fig. 10. Full visualisation of best decision tree

and a few IF-THEN rules:

Sample IF–THEN Rules:

1. IF Shucked_weight $\leq$ 0.14 AND Shucked_weight $\leq$ 0.07 AND Sex_I $\leq$ 0.29 THEN Class 0

2. IF Shucked_weight $\leq$ 0.14 AND Shucked_weight $\leq$ 0.07 AND Sex_I $>$ 0.29 AND Shell_weight $\leq$ 0.50 THEN Class 0

3. IF Shucked_weight $\leq$ 0.14 AND Shucked_weight $\leq$ 0.07 AND Sex_I $>$ 0.29 AND Shell_weight $>$ 0.50 THEN Class 1

4. IF Shucked_weight $\leq$ 0.14 AND Shucked_weight $>$ 0.07 AND Viscera_weight $\leq$ 0.50 THEN Class 1

5. IF Shucked_weight $\leq$ 0.14 AND Shucked_weight $>$ 0.07 AND Viscera_weight $>$ 0.50 AND Shucked_weight $\leq$ 0.12 THEN Class 0

C. Pruning Investigation

Table VI presents the comparison between pre-pruning and post-pruning strategies for Decision Trees.

TABLE VI
COMPARISON OF PRE-PRUNING VS POST-PRUNING DECISION TREES

Method	Train F1	Test F1	Train AUC	Test AUC
Pre-Prune	0.636 ± 0.007	0.619 ± 0.012	0.838 ± 0.004	0.819 ± 0.010
Post-Prune	0.641 ± 0.007	**0.623 ± 0.013**	0.845 ± 0.004	**0.824 ± 0.011**

Post-pruning achieves marginally higher test F1 (0.623) compared to pre-pruning (0.621). Pre-pruning constrains tree growth during training through parameters such as maximum depth and minimum samples per leaf, while post-pruning allows full tree growth before removing branches via cost-complexity pruning. We observe that pre-pruning produces slightly higher training scores but lower test scores, suggesting mild overfitting. Post-pruning exhibits higher variance across splits (test F1 std of 0.017 vs 0.009), indicating less stable performance. The difference between pruning strategies is marginal, with both methods producing comparable results on this dataset.

D. Random Forest: Effect of Number of Trees

Table VII and Figure 10 present the effect of varying the number of trees (n_estimators) in Random Forest while keeping other hyperparameters fixed.

TABLE VII
EFFECT OF NUMBER OF ESTIMATORS ON RANDOM FOREST PERFORMANCE

n_estimators	Train F1	Test F1	Train AUC	Test AUC
10	0.752	0.653	0.928	0.845
50	0.754	0.650	0.931	0.847
100	0.750	0.653	0.931	0.848
200	0.752	0.651	0.931	0.849
300	0.751	**0.650**	0.931	**0.849**

Across the range of $n_{\text{estimators}}$, we observe that the test F1 score fluctuates noticeably, with small increases and decreases rather than a monotonic trend. Test F1 reaches its highest value at $n_{\text{estimators}} = 20$ (approximately 0.653), but subsequently varies within a narrow band, declining to around 0.649 at $n_{\text{estimators}} = 300$. In contrast, the training F1 score remains relatively stable, exhibiting only minor variation and peaking at $n_{\text{estimators}} = 50$ before returning to roughly its initial level by $n_{\text{estimators}} = 300$.

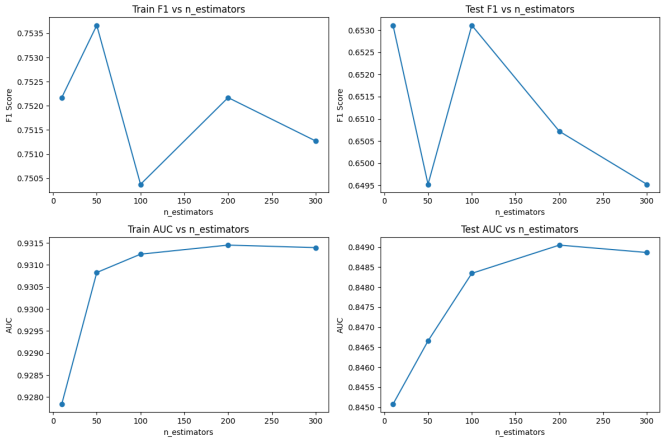

Fig. 11. Random Forest with varying number of estimators

AUC results show a clearer and more consistent pattern. Training AUC increases smoothly with more estimators, plateauing near $n_{\text{estimators}} = 200$–$300$. Test AUC follows a similar trend, rising steadily from 0.845 at $n_{\text{estimators}} = 20$ to its peak of approximately 0.849 at $n_{\text{estimators}} = 200$, with only minimal change beyond this point.

These results indicate that while F1 exhibits sensitivity to small prediction changes and does not provide a strictly increasing trend, AUC demonstrates more stable and reliable improvements with additional trees. Overall, the marginal gains diminish beyond roughly 100–200 estimators, suggesting that a moderately sized forest is sufficient for strong generalisation performance on this dataset.

E. PCA Dimensionality Reduction

Table VII presents the effect of applying PCA with different variance retention thresholds on XGBoost performance.

TABLE VIII
MODEL PERFORMANCE WITH DIFFERENT PCA VARIANCE THRESHOLDS

PCA Variance	Train F1	Test F1	Train AUC	Test AUC
95%	0.610 ± 0.004	0.592 ± 0.012	0.804 ± 0.003	0.770 ± 0.011
98%	0.612 ± 0.005	**0.593 ± 0.014**	0.807 ± 0.005	**0.771 ± 0.013**

PCA substantially reduces model performance across all metrics. XGBoost without PCA achieves test F1 of 0.647, while both PCA configurations achieve only 0.595—a decrease of approximately 5 percentage points. Test AUC drops from 0.851 to approximately 0.770, representing an 8 percentage point reduction. The 95% and 98% variance retention thresholds produce nearly identical results, suggesting that the additional 3% variance captured by the 98% threshold does not contain discriminative information for this classification task. These results indicate that PCA discards features important for distinguishing between age classes in the Abalone dataset.

F. Neural Network Experiments

1) Adam vs SGD Comparison: Table IX presents the comparison between Adam and SGD optimisers using default hyperparameters From the EDA, we selected *length* and *diameter* for the selected set.

TABLE IX
NEURAL NETWORK: ADAM VS SGD COMPARISON (50 SPLITS)

Dataset	Solver	Train F1	Test F1	Train AUC	Test AUC
unnorm_all	Adam	0.647	**0.646**	0.861	**0.858**
	SGD	0.594	0.594	0.782	0.778
unnorm_selected	Adam	0.579	**0.580**	0.753	**0.755**
	SGD	0.522	0.522	0.670	0.669
norm_all	Adam	0.644	**0.644**	0.858	**0.855**
	SGD	0.598	0.598	0.779	0.774
norm_selected	Adam	0.580	**0.581**	0.755	**0.757**
	SGD	0.563	0.563	0.712	0.710
pca_0.95	Adam	0.598	**0.596**	0.775	**0.771**
	SGD	0.595	0.595	0.761	0.758
pca_0.98	Adam	0.599	**0.596**	0.776	**0.771**
	SGD	0.595	0.595	0.761	0.758

Adam consistently outperforms SGD across all dataset configurations. On the unnormalised full feature set, Adam achieves test F1 of 0.646 compared to SGD's 0.594—a difference of over 5 percentage points. The performance gap is most pronounced on unnormalised and normalised full feature sets and diminishes when PCA is applied, where both optimisers achieve similar test F1 scores around 0.595-0.596. Adam also achieves substantially higher AUC scores, with test AUC of 0.858 versus 0.779 for SGD on the unnormalised full feature set. These results demonstrate that Adam's adaptive learning rate mechanism provides significant advantages over vanilla SGD for this classification task.

2) L2 Regularisation with Adam: Table X presents the effect of L2 regularisation (weight decay) on neural network performance using Adam optimiser.

TABLE X
NEURAL NETWORK: L2 REGULARISATION WITH ADAM - TEST F1 (50 SPLITS)

Dataset	α=0.0001	α=0.001	α=0.01
unnorm_all	**0.646**	0.646	0.646
unnorm_selected	0.580	0.580	**0.580**
norm_all	0.644	0.644	**0.644**
norm_selected	0.581	0.581	**0.581**
pca_0.95	**0.596**	0.596	0.595
pca_0.98	0.595	**0.596**	0.596

L2 regularisation has negligible effect on model performance across all tested alpha values (0.0001, 0.001, 0.01). Test F1 scores remain virtually identical regardless of the regularisation strength, with variations of less than 0.002 across all configurations. This suggests that the default MLP architecture with a single hidden layer of 100 neurons does not suffer from significant overfitting on this dataset, and thus regularisation provides no meaningful benefit. The pattern is consistent across all dataset variants, including unnormalised, normalised, and PCA-transformed features.

G. CMC Dataset

Table XI presents the 2 best models from the *Abalone* investigation that trained and tested on the *CMC* dataset. It includes the performance comparison of XGBoost and Neural Network (Adam) classifiers with Optuna-optimised hyperparameters.

TABLE XI
XGBOOST, NEURAL NETWORK WITH CMC DATASET (50 SPLITS)

Model	Train F1	Test F1	Train AUC	Test AUC
XGBoost	0.596 ± 0.009	**0.553 ± 0.031**	0.783 ± 0.005	**0.742 ± 0.025**
Neural Network	0.583 ± 0.011	0.559 ± 0.030	0.763 ± 0.008	0.739 ± 0.022

IV. DISCUSSION

A. Summary of Results

This study evaluated multiple machine learning approaches for multi-class classification on the Abalone and CMC dataset, with the goal of predicting age classes from physical measurements. Our experiments yielded several key findings.

Among tree-based models, XGBoost achieved the best performance with a test F1 score of 0.647, followed closely by Gradient Boosting (0.646) and Random Forest (0.644). The single Decision Tree underperformed all ensemble methods with a test F1 of 0.625, confirming the well-established advantage of ensemble techniques. The investigation into pruning strategies revealed that post-pruning via cost-complexity pruning achieved marginally better generalisation (test F1 of 0.625) compared to pre-pruning (0.621), though the difference was not substantial. The Random Forest experiments demonstrated that increasing the number of trees beyond 50-100 did not improve test performance and may slightly degrade it, suggesting diminishing returns from larger ensembles.

For neural networks, the Adam optimiser significantly outperformed SGD, achieving test F1 of 0.646 compared to 0.594—a difference of over 5 percentage points. L2 regularisation had negligible effect on performance across all tested alpha values, indicating that the default MLP architecture did not suffer from overfitting on this dataset.

Notably, PCA dimensionality reduction degraded performance for both XGBoost and neural networks, reducing test F1 from approximately 0.647 to 0.595. This finding suggests that PCA discards discriminative information important for this classification task.

B. Implications

Our findings align with established machine learning literature. The superior performance of ensemble methods over single decision trees is well-documented, as ensembles reduce variance through averaging (Random Forest) or iteratively correct errors (boosting methods). The comparable performance of XGBoost, Gradient Boosting, and Random Forest suggests that for tabular datasets of moderate size, the choice between these methods may be less critical than proper hyperparameter tuning.

The superiority of Adam over SGD in neural network training is consistent with prior research demonstrating that adaptive learning rate methods converge faster and achieve better optima, particularly when using default hyperparameters without extensive tuning. The negligible effect of L2 regularisation suggests that for small to medium-sized networks on this dataset, explicit regularisation may be unnecessary.

The negative impact of PCA is noteworthy. While PCA is often applied as a preprocessing step to reduce dimensionality and noise, our results demonstrate that it can remove task-relevant information. The Abalone dataset has only 9 features after encoding, and the original features (physical measurements) likely have direct interpretable relationships with the target variable. PCA's linear transformation may obscure these relationships, particularly when nonlinear classifiers like XGBoost can inherently handle feature interactions.

C. Limitations

Several limitations should be acknowledged. First, hyperparameter optimisation was performed on a single split (random_state=0) and applied across all evaluation splits. While this approach is computationally efficient, optimising on each split independently might yield different conclusions, albeit at significantly higher computational cost.

Second, the neural network experiments used only the default scikit-learn MLP architecture with a single hidden layer of 100 neurons. More complex architectures with multiple hidden layers, different activation functions, or dropout regularisation might achieve better performance and show different sensitivity to L2 regularisation.

Third, our study focused on F1-micro and ROC-AUC as evaluation metrics. Other metrics such as balanced accuracy or class-specific F1 scores might reveal different patterns, particularly given the class imbalance in the Abalone dataset.

Finally, the Abalone dataset is relatively small (4,177 samples) with a limited feature set. The conclusions drawn may not generalise to larger or higher-dimensional datasets where PCA might provide more benefit and where deeper neural networks might outperform tree-based methods.

Despite these limitations, our systematic comparison across multiple models, preprocessing strategies, and hyperparameter configurations provides valuable insights for practitioners working with similar tabular classification problems.

V. CONCLUSION

Our major project conducted a comprehensive evaluation of machine learning methods for predicting abalone age groups. By preprocessing the data, we transformed the problem into a four-class classification task, thereby developing a range of models, including pre/post-pruned Decision Trees, Random Forests, Gradient Boosting, XGBoost, and neural networks trained with Adam and SGD. We could then benchmark model performance with relevant metrics such as accuracy, F1-score, and ROC-AUC.

We performed data preprocessing to clean the dataset, such as one-hot encoding of the *sex* attribute to remove ordinal assumptions, stratified test-train splitting, feature standardisation, AND dimensionality reduction through PCA. We also provided the relevant data visualisation such as feature histograms, PCA scatter plots and tree diagrams to understand properties of feature distributions and correlation.

For all models, we examined and reported the influence of hyperparameter tuning, including tree depth and pre/post-pruning strength for Decision Trees, number of trees for Random Forests, learning rate for boosting models, and regularisation/dropout rate for neural networks, among others. To ensure the reliability of results, we ran random yet reproducible experiments with stratified test/train splits. Thus, we were able to assess the performance of each model along with the advantage and disadvantages of each.

By providing a comparative analysis of all modelling approaches and hyperparameter configurations, we were able to draw conclusions on the best combination for predictive performance. Not only this, we gained a stronger understanding of how data preprocessing and hyperparameter choices influence each model's predictive performance and generalisation.

Some directions for further research include:

1) Performing more rigorous optimisation over the hyperparameter space.
2) Running more advanced subset selection to identify the feature combination with strongest predictive performance.
3) Investigating methods to actually address the identified class imbalance and feature correlation.
4) Exploring more advanced neural network architectures and benchmark predictive performance on the abalone dataset.
5) Employing stronger validation approaches to ensure generalisation.
6) Exploring a wider range of ensemble techniques, including stacking, ExtraTrees, and more.

REFERENCES

[1] Alnuaimi, Amer; Albaldawi, Tasnim. (2024). *An overview of machine learning classification techniques*. BIO Web of Conferences. 97. 00133. 10.1051/bioconf/20249700133.

[2] Rokach, Lior; Maimon, Oded. (2005). *Decision Trees*. 10.1007/0-387-25465-X_9.

[3] Salman, Hasan; Kalakech, Ali; Steiti, Amani. (2024). *Random Forest Algorithm Overview*. Babylonian Journal of Machine Learning. 2024. 69–79. 10.58496/BJML/2024/007.

[4] Ali, Zeravan; Abuljabbar, Ziyad; Tahir, Hanan; Sallow, Amira; Almufti, Saman. (2023). *eXtreme gradient boosting algorithm with machine learning: A review*. Academic Journal of Nawroz University. 12. 320–334. 10.25007/ajnu.v12n2a1612.

[5] Islam, Mohaiminul; Chen, Guorong; Jin, Shangzhu. (2019). *An Overview of Neural Network*. American Journal of Neural Networks and Applications. 5. 05. 10.11648/j.ajnna.20190501.12.

[6] Rossi, Fábio; Caggiani Luizelli, Marcelo; Lorenzon, Arthur; Caicedo, Mauricio; Cordeiro, Weverton; Canfore, Ronaldo. (2021). *In-Network Neural Networks: Challenges and Opportunities for Innovation*. IEEE Network. 35. 7. 10.1109/MNET.101.2100098.

[7] Shanbharkar, Anurag; Rakhade, Vijay; Yadav, Lowlesh. (2024). *NEURAL NETWORK*. 10.59367/5ab5fk78.

[8] Kadhim, Zahraa; Abdullah, Hasanen; Ghathwan, Khalil. (2022). *Artificial Neural Network Hyperparameters Optimization: A Survey*. International Journal of Online and Biomedical Engineering (iJOE). 18. 59–87. 10.3991/ijoe.v18i15.34399.

[9] Irie, Kazuki; Lake, Brenden M. (2024). *Neural networks that overcome classic challenges through practice*. arXiv preprint arXiv:2410.10596.

[10] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[11] Alam, M.N.U.; Basava, K.; Chitransh, A.; et al. (2025). *Machine learning in biological research: key algorithms, applications, and future directions*. BMC Biology 23, 324. https://doi.org/10.1186/s12915-025-02424-3.

[12] N. Manjunathan, S. D, P. Girija, V. M, M. S and S. S. R. (2024). *Ensemble Learning: Importance, Applications, Challenges, and Future Work*. 2024 International Conference on Integration of Emerging Technologies for the Digital World (ICIETDW), Chennai, India. doi: 10.1109/ICIETDW61607.2024.10939877.

[13] Jerome H. Friedman. (2001). *Greedy function approximation: A gradient boosting machine*. Annals of Statistics. 29(5). 1189–1232.

[14] Nash, W.; Sellers, T.; Talbot, S.; Cawthorn, A.; Ford, W. (1994). *Abalone*. UCI Machine Learning Repository. https://doi.org/10.24432/C55C7W.

[15] Lim, T. (1999). Contraceptive Method Choice [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C59W2D.