A cover of a report

Description automatically generated

Table of Contents

[1.0 Introduction 4](#_Toc167042745)

[1.1 Background 4](#_Toc167042746)

[1.2 Objectives 4](#_Toc167042747)

[2.0 Data Visualisation and Exploration 5](#_Toc167042748)

[3.0 Data Cleaning and Preprocessing 9](#_Toc167042749)

[3.1 Removing Unnecessary Data 9](#_Toc167042750)

[3.2 Handling Missing Values 9](#_Toc167042751)

[3.3 Encoding Categorical Variables 10](#_Toc167042752)

[3.4 Feature Type Conversion 10](#_Toc167042753)

[3.5 Normalisation and Standardisation 11](#_Toc167042754)

[3.6 Data Splitting 11](#_Toc167042755)

[4.0 Decision Tree Implementation 12](#_Toc167042756)

[4.1 Classification 12](#_Toc167042757)

[4.2 Regression 12](#_Toc167042758)

[5.0 Extreme Gradient Boosting Implementation 13](#_Toc167042759)

[5.1 Classification and Regression 13](#_Toc167042760)

[6.0 Multilayer Perception Implementation 15](#_Toc167042761)

[6.1 Classification 15](#_Toc167042762)

[6.2 Regression 16](#_Toc167042763)

[6.3 Training Procedure 17](#_Toc167042764)

[7.0 Decision Trees Analysis 19](#_Toc167042765)

[7.1 Classification 19](#_Toc167042766)

[7.2 Regression 21](#_Toc167042768)

[8.0 Extreme Gradient Boosting Analysis 23](#_Toc167042770)

[8.1 Classification 23](#_Toc167042771)

[8.2 Regression 24](#_Toc167042772)

[9.0 Multilayer Perceptron Analysis 25](#_Toc167042773)

[9.1 Classification 25](#_Toc167042774)

[9.2 Regression 26](#_Toc167042775)

[10.0 Discussion 27](#_Toc167042776)

[11.0 Conclusion 28](#_Toc167042777)

[12.0 References 29](#_Toc167042778)

# 0.0 Abstract

This comprehensive report evaluates the efficacy of several machine learning algorithms, specifically decision trees, gradient boosting via XGBoost, and multilayer perceptron’s (MLPs), applied to a dataset of thorax and wing traits in Drosophila. By employing meticulous data preprocessing techniques and targeted model-specific optimization strategies, we systematically assess each algorithm's performance across a spectrum of metrics, including accuracy, mean squared error (MSE), R², and additional classification metrics. The findings reveal that while each model exhibits distinctive advantages, ensemble methods such as XGBoost consistently surpass others, demonstrating superior accuracy and robustness in predictions. This analysis not only enhances our understanding of machine learning applications within the sphere of biological data analysis but also paves the way for future innovations and methodological advancements in the field.

Notebooks: <https://github.com/adamFittlerGit/DataReport>

# 1.0 Introduction

## 1.1 Background

Drosophila melanogaster, often referred to as fruit flies, have been a focal point of genetic research and developmental biology for over a century. Their rapid life cycle and genetic tractability afford scientists unique opportunities to explore genetic phenomena and developmental processes. In modern research, the ability to accurately classify and predict phenotypic traits from genetic and physical data is crucial. Machine learning, with its capacity to manage and interpret complex datasets, is increasingly integral to these studies. However, the challenge lies in selecting and tuning the appropriate algorithm that aligns with the specific dynamics and nuances of biological data, ensuring reliable and interpretable results.

## 1.2 Objectives

This report seeks to meticulously evaluate the performance of three distinct machine learning models—decision trees, XGBoost, and MLPs—across two primary tasks: classifying the sex of Drosophila and predicting their wing loading characteristics. These models were selected to cover a broad spectrum from relatively simple decision models to more sophisticated ensemble and neural network architectures. This selection enables a comprehensive analysis of their utility in biological datasets, focusing on their effectiveness, scalability, and the clarity they bring to biological insights. Through this evaluation, we aim to identify the model that not only performs optimally in predictive accuracy but also offers the best interpretability, which is essential for further biological validation and exploration.

# 2.0 Data Visualisation and Exploration

A screenshot of a computer program

Description automatically generatedBefore diving into the models, the provided dataset was visualised and explored in order to access whether there were any specific trends or patterns in the selected tasks that could be identified. Below shows a quick summary of the features in the dataset. It can be seen that each data point is made up of 17 features, three of which being categorical, that being Species, Population and Sex. The remaining 14 numerical variables include the Latitude, Longitude, Temperature, Vial, Replicate, Thorax Length, l2, l3p, l3d, lpd, l3, w1, w2, w3 and Wing Loading.

***Figure 2.2: Feature null -values***

***Figure 2.1: Feature summary***

A screenshot of a computer

Description automatically generated

Furthermore, we can see that the dataset is made up of 1731 unique entries with only two columns having one null value each, those being the Thorax Length and Wing Loading. This is good to keep in mind when conducting the cleaning of the dataset as we know that most of the data is complete, and we do not need to worry about estimating a large quantity of the data with means and modes.

Next it is best to take a closer look at our features in order to try and identity any patterns or relationships especially in our chosen problems of sex classification and wing loading for regression. Firstly, we should further explore our categorical features.

A graph showing a number of species

Description automatically generatedBelow is a histogram for the species as well as the population features for the dataset. It can be seen that there are only two different species that this dataset explores that being Aldrich and Buzzati. These seem to be split relatively evenly between the dataset with 891 Buzzati and 840 Aldrich.

***Figure 2.3: Species count plot***

When looking at the Histogram plot of the different populations below. The dataset is made up of five unique populations that being Binjour, Gogango Creek, Grandchester, Oxford Downs and Wahruna. It is also evident here as in the species that they are relatively evenly distributed with the most frequent being Gogango Creek with 354 datapoints.

A graph showing a number of different colored rectangular objects

Description automatically generated

***Figure 2.4: Population count plot***

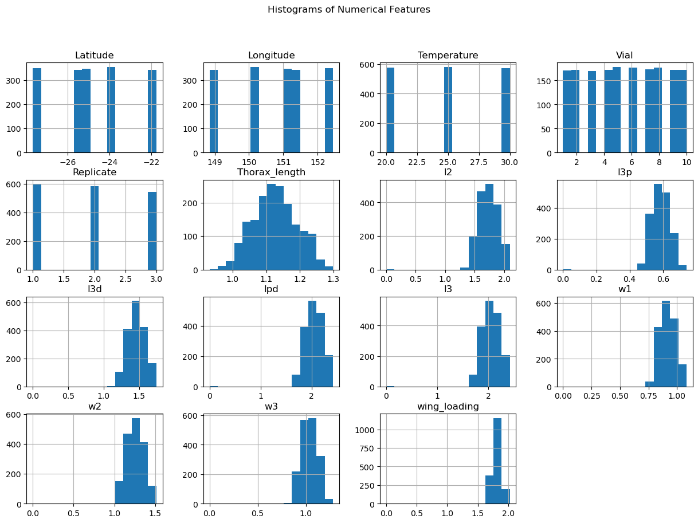
Finally for the sex of the dataset seems to also be relatively evenly distributed between the female and male populations. With slightly more males at 872 to 859, meaning that for the chosen sec classification problem this dataset should provide a good balance between male and female data hopefully resorting in less skewed models.

A blue and orange squares

Description automatically generated

***Figure 2.5: Sex count plot***

When looking at the numerical data histograms below. It can be noted that the Latitude, Longitude, Temperature, Vial, and Replicate results are separated into five distinct groupings, indicating that these variables most likely dependant on the categorical population values seen earlier, also broken up into five distinct groups. These are likely the specific latitudes and longitudes associated with each population’s areas of origin.



***Figure 2.6: Numerical Feature Histograms***

On the other hand, Thorax Length, l2, l3p, l3d, lpd, l3, w1, w2, w3 and Wing Loading are all more or less normally distributed each skewed around a different mean with varying degrees of spread. Indicating an increased degree of complexity between these features in the output.

A screen shot of a graph

Description automatically generated

***Figure 2.7: Feature Correlation Heatmap***

The correlation Heatmap above outlines the relationship between the features allowing further insight into their dependencies. It can be seen that there is a large amount f correlation between the final ten numerical features, that being Thorax Length, l2, l3p, l3d, lpd, l3, w1, w2, w3 and Wing Loading. Therefore, indication that these features are vary interdependent on each other and thus in terms of predicting the Wing Loading in our regression problems these features may tern out to have a higher degree of importance.

It is vital to note that the visualisations and explorations below while allowing insight into potential relationships are just based upon basic statistical assumptions and thus should be taken as fact, rather as suggestions or initial indications that further investigation and analysis are warranted.

# 3.0 Data Cleaning and Preprocessing

## 3.1 Removing Unnecessary Data

The first step in cleaning and processing the data for use in the models is to remove any columns or features that are unnecessary to the training of the model, in order to reduce and wasted space in computations. For this dataset each row also included the start and end year of the study.

However, as each row has the same start and end year of 1994 this feature was deemed redundant and was thus removed. The code below outlines how this can be done in place with a pandas’ data frame.



3.2 Handling Missing Values

The next step in data cleaning is to handle any missing values in the dataset to ensure that the model is able to use as many of the rows of data as possible. As discovered earlier in our exploration of our dataset, there are only two missing features in the whole dataset. One in Wing Loading and One in Thorax Length.

The general convention when handling missing value is that for numerical variables with fill the missing data with the mean of that feature. Furthermore, with categorical variables the most frequent value (mode) is used to fill in the missing data. This strategy maintains the statistical properties of the dataset without introducing significant bias.

A screen shot of a computer code

Description automatically generated

The above code was used to execute the process described above for the dataset. Ensuring there was no null columns or missing data.

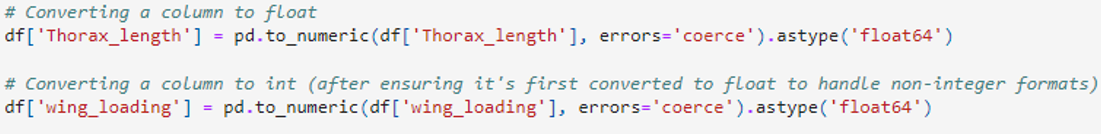
3.3 Encoding Categorical Variables  
In order to ensure that the categorical variables can be used in the model, they must be transformed into numerical values. In order to preserve their state an order o usually a label encoder or one hot encoder is used to transform the features. For this dataset the categorical variables 'Species', 'Population', and 'Sex' were transformed using label encoding. This conversion assigns each category a unique integer, facilitating the inclusion of these variables in the machine learning models.

A computer code with black text

Description automatically generated

The code above illustrates how to fit and transform the features using the label encoding. As this has been done before any train test split all of the data is not encoded and the encoders do not bee to be used again so one instance is found. If this were a production model each a separate encoder would need to be fitted and saved for each feature in order to be reused on non-encoded data in the future.

3.4 Feature Type Conversion  
When loading data into a panda’s data frame, a common issue is data types not loading in correctly which can cause errors with the modelling down the line. When this dataset is initially read into the data frame the 'Thorax\_length' and 'wing\_loading' features are given the object data type by default. In order for the model to run correctly these must be cast base to a float64 data type like the rest of the numerical variables . This ensures consistency across all features, which is crucial for effective model training and evaluation.



The code above provides an example of how this can be done using pandas. Furthermore, performing type checks and ensure all columns are cast to the correct data type is an important step in the cleaning process and should not be overlooked.

3.5 Normalisation and Standardisation  
In this project, all numerical features were standardised using the Standard Scaler (values scaled with mean 0 and standard deviation 1). This preprocessing step is crucial to mitigate the impact of scale differences across features. Standardisation helps prevent model bias towards variables measured on larger scales, ensuring that each feature contributes equally to the model training process.

Standardised features are also more interpretable, as the coefficients in the model represent the change in the output for a one-standard-deviation change in the input. This preprocessing step is particularly useful for algorithms that assume normally distributed features or require standardised features for optimal performance, particularly for the neural networks used later.

A screen shot of a computer code

Description automatically generatedA screen shot of a computer code

Description automatically generated

Here is the code outlining how the scalar is used on the dataset for the classification and the regression problem after the labels have been split from the features.

## 3.6 Data Splitting

In order to estimate the performance of the model on new, unseen data (Enew), the dataset was split into training and holdout sets. The holdout allows for the estimation of Enew using Ehold-out. Thus, the performance of the model on the unseen data in the test set should provide a good approximation for its performance on new data.

This estimation is crucial for understanding the model's ability to generalise and provides valuable insights into its expected performance in practical applications.





The code snippets above outline a train test split for 90/10 for the dataset for both classification and regression. Note for classification stratification is used for the label in order to ensure that the training and test splits both contain an amble amount of each class in order to avoid a scenario where one class in only found in either the train or test set.

# 4.0 Decision Tree Implementation

## 4.1 Classification

For classification, the “DecisionTreeClassifier” from scikit-learn was used. Furthermore, in order to ensure the best possible hyperparameter, grid search cross validation is employed. Thus, the parameters for the classifier were defined and the range for each to be search is provided. These parameters for the decision tree where the depth, with the range of 3,5,10 as the options for the optimisation. The minimum samples for the splits and leaves were also provided in the ranges of 2, 10, 20 and 1, 5, 10 respectively. These parameters allow the model to be regularised. Finally, the criterion could be either Gini index or Entropy depending on which one causes the model to perform better. Finally grid search is fit the model to the data as well as performing 10-fold cross validation. The model is scored on the accuracy of its classification. The code snippet below highlights this process.

A computer code with black text

Description automatically generated

## 4.2 Regression

The DecisionTreeRegressor was used for the regression task, with similar hyperparameter tuning using grid search cross validation. Here the regression model also includes regularisation through the use of the min samples split and leaf learned parameters. It also adds another parameter for the max features electing to either use the log2 or sqrt of the total number of features. Finally, this model also includes the 10-fold cross validation within the grid search and uses negative mean squared error in order to score its accuracy.

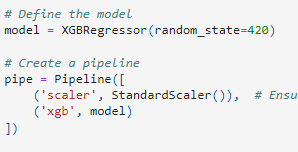
A computer code with text

Description automatically generated with medium confidence

# **5.0 Extreme Gradient Boosting Implementation**

## 5.1 Classification and Regression

For classification the XBGClassifier was used as the base model and XGRegressor was used for the regression task. In order to ensure all of the hyperparameters were effectively and efficiently encapsulated into a single object Sklearn’s pipeline object was utilised. This allows for the model and the scaler to be grouped together ensuring that the optimiser will have all that it needs in order to train the model.

A screen shot of a computer code

Description automatically generated

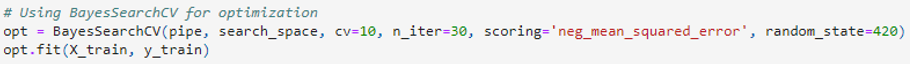
In order to get the most out of the model, each of the specific hyperparameters associated with the XGBoost model needed to be optimised. For this Bayesian Search optimisation was chosen over traditional methods such as grid search or randomised search. This is because Bayesian search uses probabilistic model in order to the predict the parameters based upon previous evaluations of them. This results in a more computationally efficient optimisation, which is vital when using a more computationally complex.

A screenshot of a computer code

Description automatically generated

The search space above specifies the range to search for each of the provided XGBoost hyperparameters, these include the max depth of each tree which was specified between 2 and 17 as 17 is the max number of features that can be used. Next the learning rate of the model was chosen to be between 0.001 and 1. Furthermore, the subsample is a boosting parameter used to decide the percentage of observations within each model’s bootstrapped dataset, here the range was chosen to be between 50% to 100% of the datapoints.

The cosample parameters for tree level and node specify the percentage of features to be used at each of those levels in order to ensure the generality of the boosting model.

The next two parameters are used to implement L1 and L2 regularisation into the model ensuring the model finds a balance in its use of the features in order to find the solutions as well as identifying which of the features are the most important in the selection. The gamma parameter also acts as a form of regularisation used to control the loss reduction threshold for splitting the tree and is used to prevent overfitting.

Finally, the number of estimators is chosen between 100 and 500 for the boosting model in order to define range for the number of successive trees that should be used to find the best solution. Furthermore, as seen below the Bayes search also incorporates a 10-fold cross validation and uses ROC\_AUC in order to judge classification accuracy and used negative mean squared error for regression accuracy.

# **6.0 Multilayer Perception Implementation**

## 6.1 Classification

For this report implementation of neural network in classification and regression, Pytorch was chosen as the main library as it allows more control over specific details of the layers when compared to the more black box approach provided by Sklearn.

Due to this in order for the dataset to be used it must first be transformed into a tensor object, this must be done for both the inputs and the outputs as seen below

A black text on a white background

Description automatically generated

In creating the classification model, it had to be considered how to deal with the issue of having only 17 input features. As the data is tabular, both the classification and regression utilised a standard fully connected architecture. Therefore, it was decided to get the most out of this architecture that the input features should first be upscaled to 128 in order to allow the system more learnable parameters and thus allow more depth in the architecture as opposed to the initial 17 features.

A screenshot of a computer program

Description automatically generatedThe coded snippet here shows the initialised architecture for the classifier. It is made up of four hidden layers initially up sampling the 17 input features into 128 and then down sampling from there to 64, 32, 16 and 1 for the final output. Each layer incorporates batch normalisation after up or down sampling to reduce computational complexity and ensure efficiency as well as a Relu activation function to add nonlinearity into the system. Finally, each layer has a dropout percentage varying between 20% and 30% to ensure the model does not become too reliant on any one feature in order to improve generality and reduce the chance of overfitting (lower the variance). Note the final layer is fun through a sigmoid function so that the output is either 0 or 1. A computer code with black text

Description automatically generated

The code above shows how the layers are pieced together during the forward step as well as how the data point x transverses through the network from the input to the output prediction.

## 6.2 Regression

A screenshot of a computer program

Description automatically generatedThe regression model to the left also follows the same basic architecture as the classification network, with a few key differences.

The main difference is that the goal here is not to classify a binary category but to predict a continuous numerical value. Therefore, the final linear layer does not require the use of the sigmoid function.

A computer code with black text

Description automatically generated

Again, the code snippet below shows the feed forward method, highlighting how the layers of the network are pieced together as well as how a data point flows through the network.

## 6.3 Training Procedure

A screenshot of a computer program

Description automatically generatedThe classification and regression models use the same procedure for training. The training phase consists of setting the model to train mode, then unwrapping the inputs and labels from the training data loader. Then the optimizers gradients are reset before calculating the model outputs and calculating the loss. The loss is then backpropagated through the network then the optimizer steps forward.

A screenshot of a computer code

Description automatically generatedFor evaluation and validation, the model is set to eval mode, next torch is set to not calculate gradients to reduce the computational complexity. Then the loss is calculated on the validation set. Finally, the scheduler is stepped forward making sure the learning rate is adjusted to attempt a hone into a local minimum.

A screenshot of a computer code

Description automatically generatedFurthermore. Like the previous model the neural network also used 10-fold cross validation in order to find the best possible train test split in order to produce the most accurate model.

A screenshot of a computer program

Description automatically generatedThe following outlines the specific setup and training for each of the folds in the cross validation. First the datasets and data loaders for training and validation are setup in order to be used later in the training.

The model is then defined, binary classifier for classification and regressor for regression. The optimizer chosen for the models was the Adam optimizer, this is commonly used over other more traditional optimizers such as gradient descent. This is because it combines adaptive learning rates for each of its parameters as well as bias corrections measurement for the calculations of the first and second gradients in order to improve convergence.

Finally, a learning rate scheduler is used in order to ensure the model parameters are adjusted at rates relative to their position from a local minimum for the model’s accuracy.

# 7.0 Decision Trees Analysis

## 7.1 Classification

The Decision Tree Classifier demonstrated robust performance with a test accuracy of 94.2%. In-depth evaluation revealed precision metrics of 0.99 for class 0 (female) and 0.91 for class 1 (male), coupled with recall rates of 0.90 and 0.99, respectively. These metrics underscore the model’s high specificity in predicting females and excellent sensitivity in predicting males. The balanced F1-score of 0.94 for both classes reflect a well-tuned model that effectively manages the trade-off between precision and recall.

Furthermore, the graph below outlines the importance of each feature in the output for the best trained model. It can be seen that the most important feature identified by the decision tree in classifying the sex of the observation was the length of the thorax.

A graph with blue bars

Description automatically generated

***Figure 7.1.1: Feature importance***

This is followed by temperature and l3 as the second and third most important features. The graph also provides insight into how the L1 and L2 regularisation affected the model. As it is evident that the model utilised most of the features to some degree while also identifying the strongest of them.

The feature importances are further reinforced in the decision tree visualisation show in Figure 7.1.2. Indicating that at the top three levels the most important features are indeed the thorax length as well as the temperature .

Overall, the results of this classifier will act as the base in which the following two classifiers will be compared. It has shown excellent performance as well as insight into the importance of each feature.

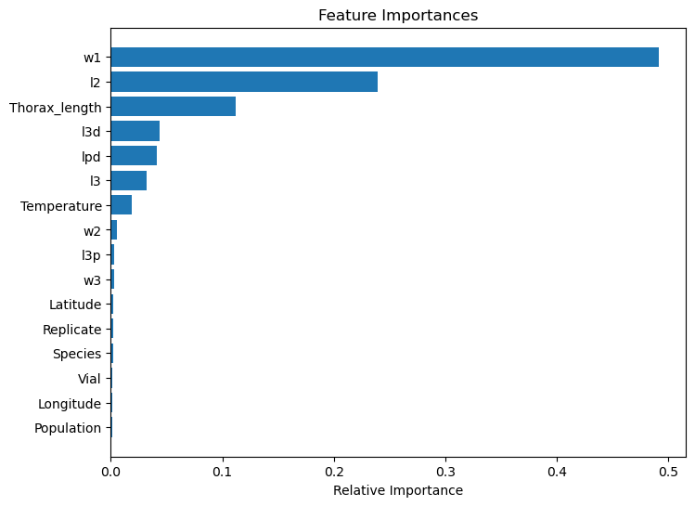
## A diagram of a diagram Description automatically generated

***Figure 7.1.2: Decision Tree Classifier best model visualisation***

## 7.2 Regression

The regression model exhibited a mean squared error (MSE) of 0.000835 and an R² value of 0.818. While these figures indicate a competent model, they also suggest potential areas for improvement in model precision and the ability to capture greater variance within the dataset.

A line graph with blue dots

Description automatically generatedSimilar to the classification model the graph to the left shows the importance of each feature to the final classification of the best model. It is evident that the most important feature identified for predicting the wing loading is the w1 feature. This is followed by l2 and finally thorax length. These are reinforced in the visualisation seen in Figure 7.2.4 This model also seems to be more persuaded by the L1 regularisation as many of the supporting features have been pushed to 0 in terms of importance in the final classification.

***Figure 7.2.1: Feature Importance***

A graph of a distribution of residuals

Description automatically generatedFurthermore, the next figure shows the relationship between the predicted outputs and the real outputs on the test dataset. Here we can see that there is a relatively positive linear trend with some variance. This is well reflected by the R2 value of 82% indicating the accuracy of the model. This coupled with the relatively low MSE of 0.00084, reinforcing the model’s generalisability on new data.

***Figure 7.2.2: Test set performance***

Finally, the residual distribution appears to be normally distributed with mean around 0. Therefore, indicating that the model is balancing underfitting and overfitting well and is not leaning in either direction. Thus, indicating a constant variance among the data.

***Figure 7.2.3: Test residual distribution***

## A diagram of a diagram Description automatically generated

***Figure 7.2.4: Decision Tree Regressor best model visualisation***

# **8.0 Extreme Gradient Boosting Analysis**

## 8.1 Classification

A graph with different colored bars

Description automatically generatedThe XGBoost classifier achieved an exemplary test accuracy of 98.1% and an ROC-AUC score of 0.98, demonstrating the model’s superior predictive power and its exceptional capability in distinguishing between classes under varying threshold settings. The high accuracy and AUC scores were complemented by impressive training and validation scores, highlighting the model's consistent performance across different subsets of data.

***Figure 8.1.1: Feature Importance***

The graph above outlines the feature importance for the best XGBoost classifier. It is immediately evident that this model compared to the decision tree classifier this model uses a much wider array of features when making its classification. As seem above this model did not find thorax length and temperature to be the most important features but it actually finds the top three features to be l3, w2 and lpd.

The wider array of features used in order to make the classifications indicates the model may have been greater influenced by the L2 regularisation terms. However, as a few of the features have been driven to 0 influence over the final result. Thus, the model seemed to get the best of both forms of regularisation, the removal of unimportant features and emphasis of important ones from L1 regularisation and lack of reliance on a single feature provided by the L2 regularisation term.

Thus, the model seems to exhibit low bias as well as low variance. Reinforcing the strength of ensemble methods such as boosting in reducing both the bias and the variance of the weak base learner.

## 8.2 Regression

The XGBoost regression model reported an outstanding R² of 0.983 and an MSE of 0.000074, indicating the model’s exceptional accuracy in predicting wing loading. This performance signifies a near-perfect predictive model that closely mirrors the observed data variations and trends.

A graph with colorful bars

Description automatically generatedFurthermore, the feature importance graph here resembles the same structure as that in the initial decision tree, exhibiting signs of the L1 regularisation with higher importance placed on a select few features when deciding the output of the model. Unlike the initial decision tree however the specific features that have been found to be the most important were lpd, l3 and temperature.

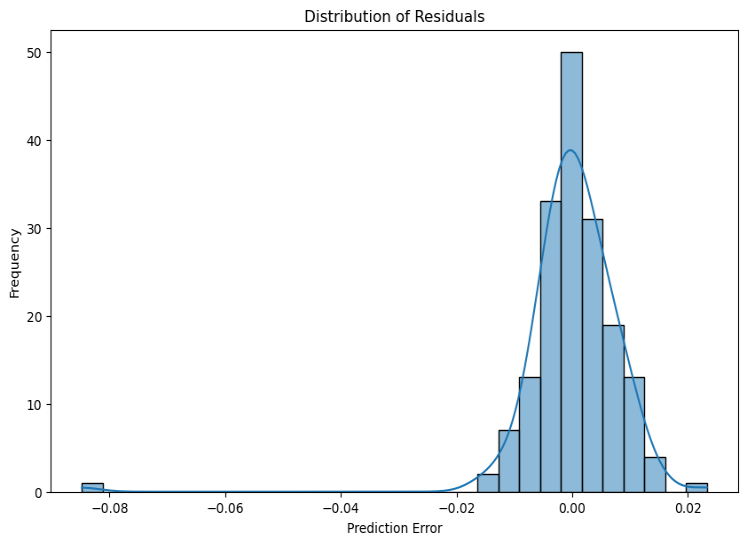
***Figure 8.2.1: Feature Importance***

***Figure 8.1.1: Feature Importance***

A graph of a graph showing the value of a value

Description automatically generated with medium confidenceWhen looking at the graph of the predicted versus the actual test outputs it is clear that the model has both low bias as well as low variance. This is reinforced by the high regression coefficient as well as the low MSE. However, it should be noted that there is an outlying point in the data with a much higher error than the rest.

***Figure 8.2.2: Test performance***

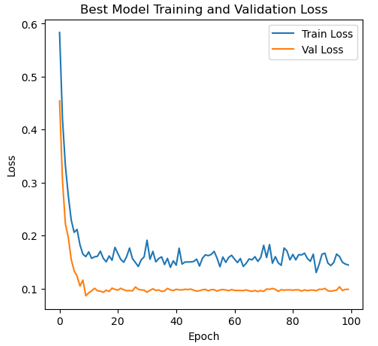
In terms of the residual errors, clearly represents an ideal normal distribution with mean 0, other than the outlying point. This indicates that the model does not tend to either underfit or overfit. This error distribution is much less skewed than that seen in the decision tree, reinforcing that the ensemble method has decreased the model’s variance.

***Figure 8.2.3: Test residual distribution***

# 9.0 Multilayer Perceptron Analysis

## 9.1 Classification

The classifier attained a high-test accuracy of 95.98%, which was achieved after rigorous tuning of the network architecture—including the number of hidden layers, the inclusion of batch normalization, and dropout settings. This process was essential to refine the model’s capacity to generalize beyond the training data without succumbing to overfitting.

When compared to the results of the decision tree classifier it has a higher accuracy on the training set, indicating both a lower bias and variance of the model. This is expected due to the less rigid structure of the neural net. However, this still lags behind the accuracy seen in the XGBoost model.

A graph of a graph

Description automatically generatedThe figure to the left outlines the training and validation loss of the best model over each epoch. It can be seen that the training loss seemed to have much more variance in its loss as opposed to the validation set over time.

***Figure 9.1.1: Training and validation loss***

This trend can also be seen when looking at Figure 9.1.2, the training accuracy seen to have much more variation as it converges as opposed to the validation set.

These Figures also indicate that the model made no significant differences in performance in terms of bias or variance after the 20th epoch. Thus, in order to further improve the model. Increasing the number of epochs will be ineffective and achieving a higher accuracy will be the result of creating a better architecture.

***Figure 9.1.2: Training and validation accuracy***

***Figure 9.1.2: Training and validation accuracy***

## 9.2 Regression

The regression model produced an R² of 0.867 and an MSE of 0.000572. While these results confirm the MLP’s capability to model complex nonlinear relationships in the data, they also highlight its relative underperformance compared to the XGBoost model, especially in terms of explaining the dataset's variance.

A graph of a graph

Description automatically generatedIt does however perform better than the decision tree regressor in terms of accuracy of the test set. Indicating slightly lower levels of bias and variance which is to be expected.

The training and validation loss reflect similar trends as seen in the classifier indicating minimal movements in the training loss after 20 runs through the dataset.

A graph with blue dots and red line

Description automatically generatedFurthermore, when looking at Figure 9.2.2 outlining the relationship between the actual and predicted outputs on the test set, it is clear that the model does perform reasonably well although the spread of the seems slightly less balanced in terms of under and overfitting.

***Figure 9.2.1: Training and validation loss***

This idea is reinforced when looking at the final figure showing the distribution of the residual errors on the test set.

***Figure 9.2.2: Test set accuracy***

A graph of a distribution of residuals

Description automatically generatedWhen compared to the relevant graphs for the other regression models it is evident that these errors are more skewed and less evenly spread while still being relatively normally distributed around 0. The distribution seems to see more errors in terms of the model underfitting while the greatest errors occurred when the model overfit. Thus, indicating it has higher variance, potentially due to overfitting the training set

***Figure 9.2.3: Test residual distribution***

***Figure 9.2.3:***

# 10.0 Discussion

Overall, all of the model performed well on the dataset in both classification and regression tasks. This is likely due to the application of multiple machine learning techniques such as: data cleaning, normalisation, standardisation, cross validation, regularisation (L1 and L2), hyperparameter optimisation and other mainly related to Neural Networks. Thus, each of the model was performing optimised to the max in order to even the playing field and determine beyond those metrics, which of the underlying strategies worked the best given the provided dataset in both classification and regression.

It was initially assumed that the simplicity of a decision tree when compared to the complexity of the other model would cause the model to lag behind in terms of both bias and variance. Although it was the weakest of the three in both classification and regression it was also the fastest to train due to its simplicity. Furthermore, while it did lag behind the results of the XGBoost models, it stood its ground well against the neural net. Given that it was much simpler and faster to train this does imply that for simpler tasks and datasets such as this one the use of a decision tree would allow one to compete with the accuracy of a neural net with much less computational resources.

In terms of the XGBoost gradient boosting models, they performed the best on all of the tasks with an accuracy of 98.1% for classification and 98.3% for regression on new data. While all of the models achieved 94%+ on the classification task, neither the decision tree nor the neural network achieved more than 87% on the regression task. Therefore, highlighting the strength provided by the boosting of features and datapoints in the iterative approach of the XGBoost models.

Finally, the Neural Network models did show higher accuracy on the test dataset when compared to the decision tree for both classification and regression. However, the models were still outmatched by the ensemble based XGBoost. Although due to the flexible and highly customisable nature of deep learning there could well have been another form of neural network architecture that may have produced better results than that of the boosting models. This is a point that could be further explained, in terms of pitting boosting against deep learning in a future report.

# 11.0 Conclusion

In conclusion, the comparative analysis of Decision Tree, XGBoost, and MLP models revealed distinct strengths and weaknesses in both classification and regression tasks. The XGBoost model emerged as the top performer, with high test accuracy, low bias, and low variance. The Decision Tree provided a solid, interpretable baseline, while the MLP demonstrated the ability to flexibly capture complex nonlinear relationships.

# 12.0 References

*Adam optimisation* . (2021). Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

*Data Visualisation*. (2024). Retrieved from Ibm: https://www.ibm.com/topics/data-visualization#:~:text=IBM-,What%20is%20data%20visualization%3F,that%20is%20easy%20to%20understand.

*Feature Selection*. (2024). Retrieved from Heavy AI: https://www.heavy.ai/technical-glossary/feature-selection#:~:text=Feature%20selection%20is%20the%20process,of%20datasets%20continue%20to%20grow.

*Gradient Descent* . (2023). Retrieved from Built In : https://builtin.com/data-science/gradient-descent

*Guide to effective data cleaning* . (2024). Retrieved from Tableau : https://www.tableau.com/learn/articles/what-is-data-cleaning

*Hyperparameter Tuning* . (2024). Retrieved from Medium : https://towardsdatascience.com/bayesian-optimization-for-hyperparameter-tuning-how-and-why-655b0ee0b399

*Model selection, Preprocessing*. (2024). Retrieved from Scikit-Learn: https://scikit-learn.org/stable/

*Neural Networks*. (2024). Retrieved from Pytorch: https://pytorch.org/tutorials/beginner/basics/buildmodel\_tutorial.html

*What is a decision tree*. (2024). Retrieved from ibm: https://www.ibm.com/topics/decision-trees#:~:text=A%20decision%20tree%20is%20a,internal%20nodes%20and%20leaf%20nodes.

*XGBClassifier, XGBRegressor*. (2022). Retrieved from XGBoost Documentation: https://xgboost.readthedocs.io/en/stable/