Classifying birds into ecological groups by bone measurement

Isabella Hofstede

2023-11-11

Isabella Hofstede
398319
BFV3
2023-11-11
Dave Langers

Bird bone measurements

Analysis of bird bone measurement as relating to species, using machine learning

Student: Isabella Hofstede
Bio-informatica
Student number: 398319
Life science en technology
Lecturer: Dave Langers
Date: 2023-11-11

Contents

1 Introduction

Birds play an important role in every ecosystem and their adaptations are often well-studied. Different types of bird species adapt to different environments. Their attributes are tied to the environments they live in, this includes the structure of their body. Can we infer the potential link between bone measurements and bird's ecological groups? This study aims to seek that link with the help of machine learning algorithms. Using a dataset from Kaggle containing measurements of bird skeletons, we explore the relation between bone structure and ecological behavior.

1.1 Research question

Which brings us to the research question; Using machine learning algorithms, which bone measurements are most informative for classifying birds into ecological groups?

2 Materials

In this study, we utilized a dataset obtained from Kaggle, specifically the "Birds, Bones, and Living Habits" dataset /cite{bird}. The dataset contains bird specimens from the collections of the Natural History Museum of Los Angeles County. Each bird in the dataset is represented by 10 measurements related to bone lengths and widths, alongside a categorical attribute known as "type," used for classifying birds into ecological groups; Swimming Birds, Wading Birds, Terrestrial Birds, Raptors, Scansorial Birds and Singing Birds.It is a 420x10 size continuous values unbalanced multi-class dataset, meaning there are 420 birds contained in this dataset, each bird is represented by 10 measurements (features):

Bone Abbreviation	**Description**
Huml	Humerus Length
Humw	Humerus Width
Ulnal	Ulna Length
Ulnaw	Ulna Width
Feml	Femur Length
Femw	Femur Width
Tibl	Tibiotarsus Length
Tilw	Tibiotarsus Width
Tarl	Tarsometatarsus Length
Tarw	Tarsometatarsus Width

Table 1: Bone Abbreviations and Descriptions

The class attribute is "type", referring to the ecological group. The bird skeletons belong to 21 orders, 153 genera, 245 species. Each bird has a label for its ecological group:

Abbreviation	**Description**
SW	Swimming Birds
W	Wading Birds
T	Terrestrial Birds
R	Raptors
P	Scansorial Birds
SO	Singing Birds

Table 2: Bird Abbreviations and Descriptions

A classifier model was used to predict the type of bird based on these bone measurements. With the help of Weka version 3.8.6 (stable) the best fitting model was chosen for the classification problem. [3]. Weka hosts a multitude of machine learning algorithms to be used. After having determined the best fitting algorithm for

this particular dataset, Java was used to make the chosen model easily distributed to the public [4]. This was done by crating a Java wrapper in the IntelliJ IDEA 2023.2. IDE [5].

3 Methods

Exploratory data analysis (EDA) was performed in Rstudio. Data pre-processing involved identifying and addressing missing values. Seven birds in the dataset had missing measurements for certain bones, accounting for less than 2% of the dataset. Given their random distribution and negligible impact, the decision was made to remove entries with missing values entirely. The ourliers however, were kept in the dataset because it made statistical sense. They were logtransformed to check for their significance. For example the SW (Swimming Bird) category is the biggest group by a big margin, this is because swimming birds have a huge range in bone measurements. To name an example, swans and little grebes both belong to the SW category, but differ in size. Swans are comparatively 10x the size of little grebes. For this reason outliers were kept in the dataset.

Since the dataset is unequally distributed, contains outliers and considerably small, the choice was made not to oversample to make sure there is no risk of overfitting and increasing the sensitivity to outliers. When dealing with imbalanced classes, making sure to choose the right evaluation metrics is critical. For this reason the focus was on precision, recall, F1 score and area under the curve. Lastly, extra care was given to choose algorithms that avoid the the aforementioned problems completely. All algorithms will be validated by using 10-fold crossvalidation.

All the visualizations of plots and tables of data were done by using the R libraries; ggplot2, GGally, GridExtra, pander and kableExtra. Everything, including the EDA, Java wrapper, data and code for this document are available on https://github.com/ishofstede/thema09 for review.

4 Results

4.1 Univariate analysis

4.1.1 Summary statistics

Comprehensive exploration of the data was conducted and the results of that will now be discussed. Firstly the nature of our data.

```
## ### Data Frame Summary
## #### overview
## **Dimensions:** 420 x 11
## **Duplicates:** 0
##
## +-------------+--------------------------+---------------------+---------+
## | Variable    | Stats / Values           | Freqs (% of Valid)  | Missing |
## +=============+==========================+=====================+=========+
## | huml\       | Mean (sd) : 64.7 (53.8)\ | 407 distinct values | 1\      |
## | [numeric]   | min < med < max:\        |                     | (0.2%)  |
## |             | 9.8 < 44.2 < 420\        |                     |         |
## |             | IQR (CV) : 65.1 (0.8)    |                     |         |
## +-------------+--------------------------+---------------------+---------+
## | humw\       | Mean (sd) : 4.4 (2.9)\   | 321 distinct values | 1\      |
## | [numeric]   | min < med < max:\        |                     | (0.2%)  |
## |             | 1.1 < 3.5 < 17.8\        |                     |         |
## |             | IQR (CV) : 3.6 (0.7)     |                     |         |
## +-------------+--------------------------+---------------------+---------+
## | ulnal\      | Mean (sd) : 69.1 (58.8)\ | 398 distinct values | 3\      |
## | [numeric]   | min < med < max:\        |                     | (0.7%)  |
## |             | 14.1 < 43.7 < 422\       |                     |         |
## |             | IQR (CV) : 69.5 (0.9)    |                     |         |
## +-------------+--------------------------+---------------------+---------+
## | ulnaw\      | Mean (sd) : 3.6 (2.2)\   | 308 distinct values | 2\      |
## | [numeric]   | min < med < max:\        |                     | (0.5%)  |
## |             | 1 < 2.9 < 12\            |                     |         |
## |             | IQR (CV) : 2.9 (0.6)     |                     |         |
## +-------------+--------------------------+---------------------+---------+
## | feml\       | Mean (sd) : 36.9 (20)\   | 402 distinct values | 2\      |
## | [numeric]   | min < med < max:\        |                     | (0.5%)  |
## |             | 11.8 < 31.1 < 117.1\     |                     |         |
## |             | IQR (CV) : 25.8 (0.5)    |                     |         |
## +-------------+--------------------------+---------------------+---------+
## | femw\       | Mean (sd) : 3.2 (2)\     | 290 distinct values | 1\      |
## | [numeric]   | min < med < max:\        |                     | (0.2%)  |
## |             | 0.9 < 2.5 < 11.6\        |                     |         |
## |             | IQR (CV) : 2.4 (0.6)     |                     |         |
## +-------------+--------------------------+---------------------+---------+
## | tibl\       | Mean (sd) : 64.7 (37.8)\ | 405 distinct values | 2\      |
## | [numeric]   | min < med < max:\        |                     | (0.5%)  |
## |             | 5.5 < 52.1 < 240\        |                     |         |
## |             | IQR (CV) : 46.5 (0.6)    |                     |         |
## +-------------+--------------------------+---------------------+---------+
## | tibw\       | Mean (sd) : 3.2 (2.1)\   | 288 distinct values | 1\      |
## | [numeric]   | min < med < max:\        |                     | (0.2%)  |
## |             | 0.9 < 2.5 < 11\          |                     |         |
```

	IQR (CV) : 2.7 (0.7)			
tarl\ [numeric]	Mean (sd) : 39.2 (23.2)\ min < med < max:\ 7.8 < 31.7 < 175\ IQR (CV) : 27.2 (0.6)	409 distinct values	1\ (0.2%)	
tarw\ [numeric]	Mean (sd) : 2.9 (2.2)\ min < med < max:\ 0.7 < 2.2 < 14.1\ IQR (CV) : 2.1 (0.7)	279 distinct values	1\ (0.2%)	
type\ [character]	1\. P\	38 (9.0%)\	0\ (0.0%)	
	2\. R\	50 (11.9%)\		
	3\. SO\	128 (30.5%)\		
	4\. SW\	116 (27.6%)\		
	5\. T\	23 (5.5%)\		
	6\. W	65 (15.5%)		

Our data consists of one class attribute type with all other attributes being numeric in nature. Type relates to the ecological group of bird and the remaining attributes are different bones, measured in milimeter. Table **??** shows that over half of the data is attributed to songbirds (SO) with 30.5% and swimming birds (SW) with 27.6%. With terrestrial birds (T) and scansorial birds (P) together making up only 14.5% of the data. Thus this is an unbalanced multi-class dataset.

4.1.2 Class distribution

The summary already showed that there is an unequal distribution, this will be visualized.

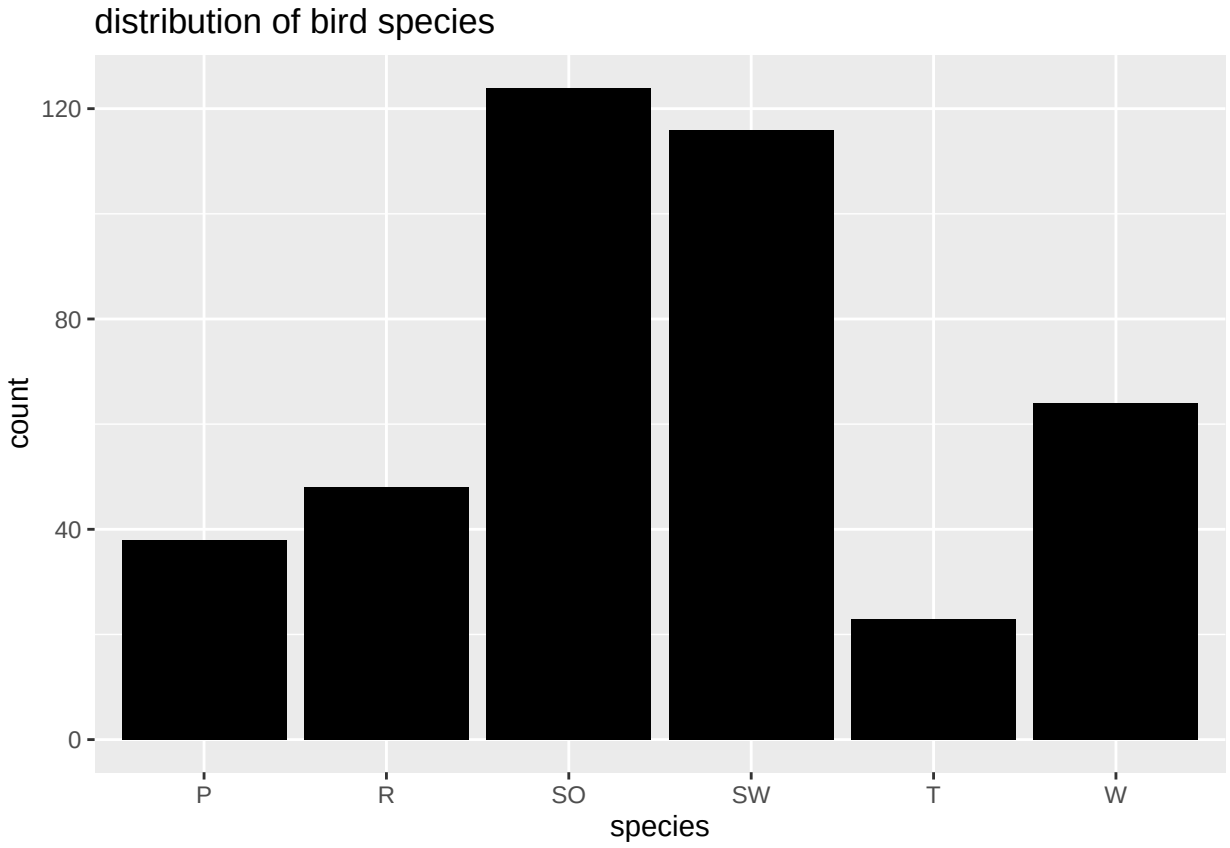


Figure 1: Distribution of bird each species

Figure 1 shows the distribution of birds is unequal. With SO and SW being the biggest groups with over 100 instances. The smallest one being T with below 30 instances.

4.2 Bivariate

4.2.1 Variation

This section will be comparing the distribution of different groups or categories. Boxplots have been made to identify the central tendency and spread of the dataset.

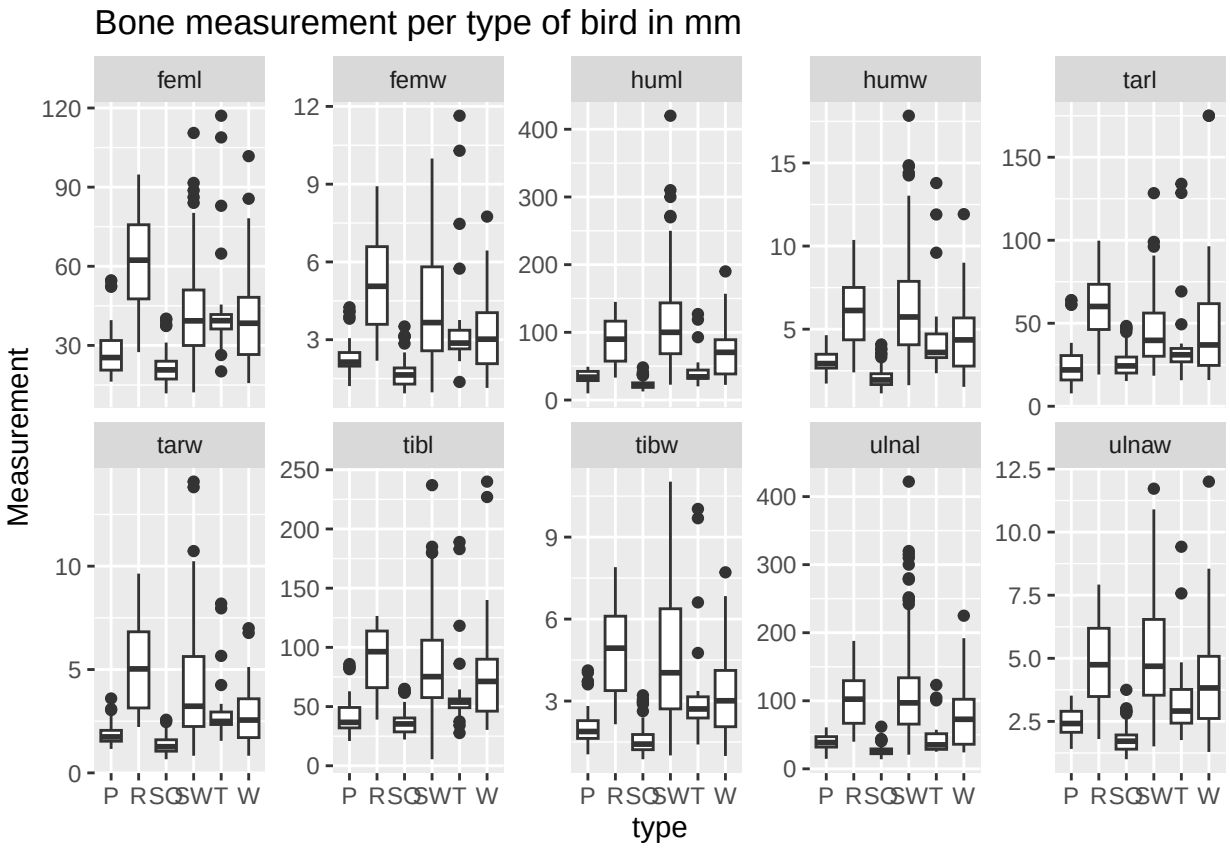
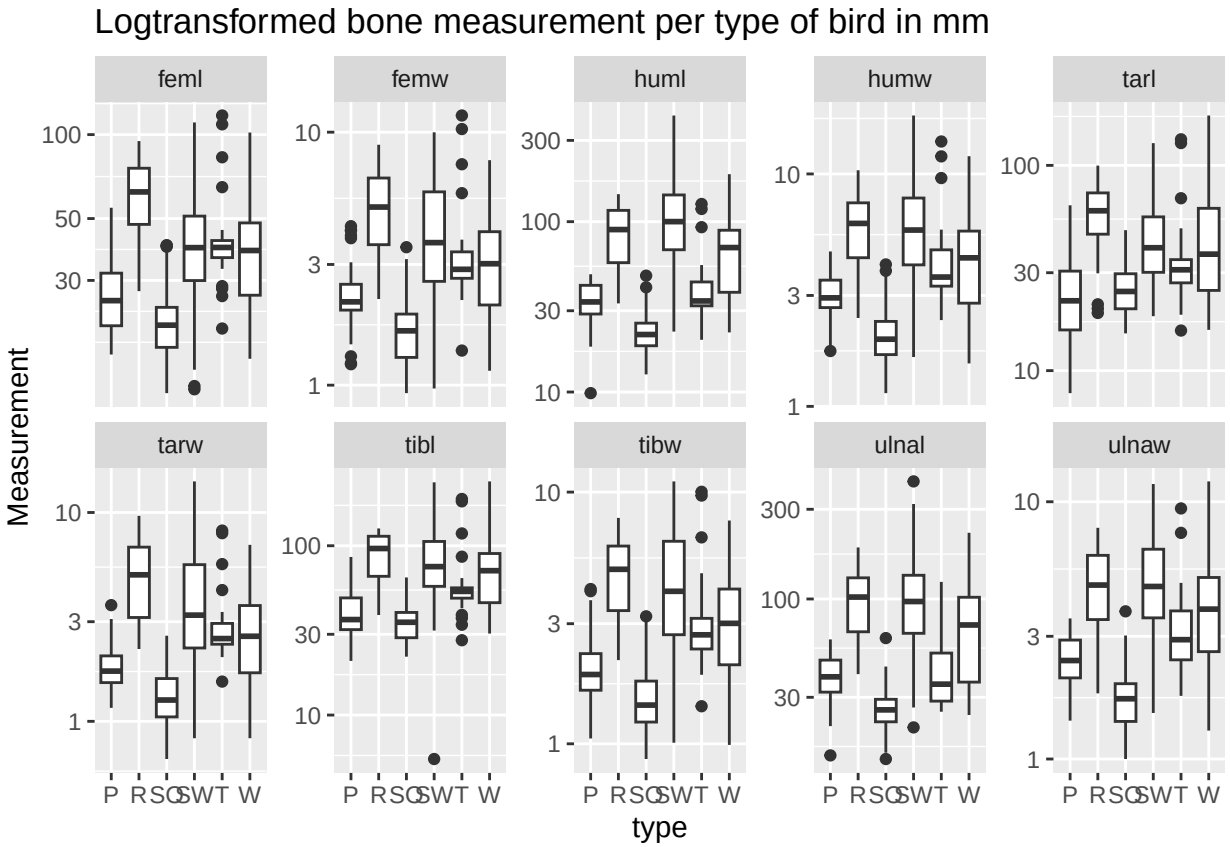


Figure 2: Values of bones (in mm) plotted for each type of bird, repeated for every attribute, the length and width of every bone.

Figure Figure~**??** shows a lot of outliers on all bird types. Because the dataset has a scewed distribution, the differences can be more interpretable on a logarithmic scale.



The log-transformed boxplots in figure Figure~**??** reveals patterns that were less apparent in the regular boxplots. After the log transformation the majority of the outliers of the biggest two classes SW and SO have been removed entirely. The outliers in the classes with the lowest amount of instances have increased.

4.2.2 Class labels

To find out which attribute would be most informational, an anova test has been performed on the attributes.

Table 3: missing values per type

Bone	p_value
huml	1.023e-49
ulnaw	1.244e-48
feml	8.762e-46
tibw	1.785e-44
humw	2.766e-44
femw	6.326e-42
tarw	4.201e-40
ulnal	2.554e-39
tibl	2.366e-36
tarl	3.724e-24

Table **??** shows the p-measurments, sorted from low to high. The lowest p-value tells us which measurement will be most informative for machine learning. It seems that huml and ulnaw have the lowest p-values, these will be most informative. The p-values of huml and ulnaw are the lowest and thus most informative.

4.3 Multivariate

4.3.1 Correlation

To see if the data is correlated, and if so, by how much; a few correlation plots were created.

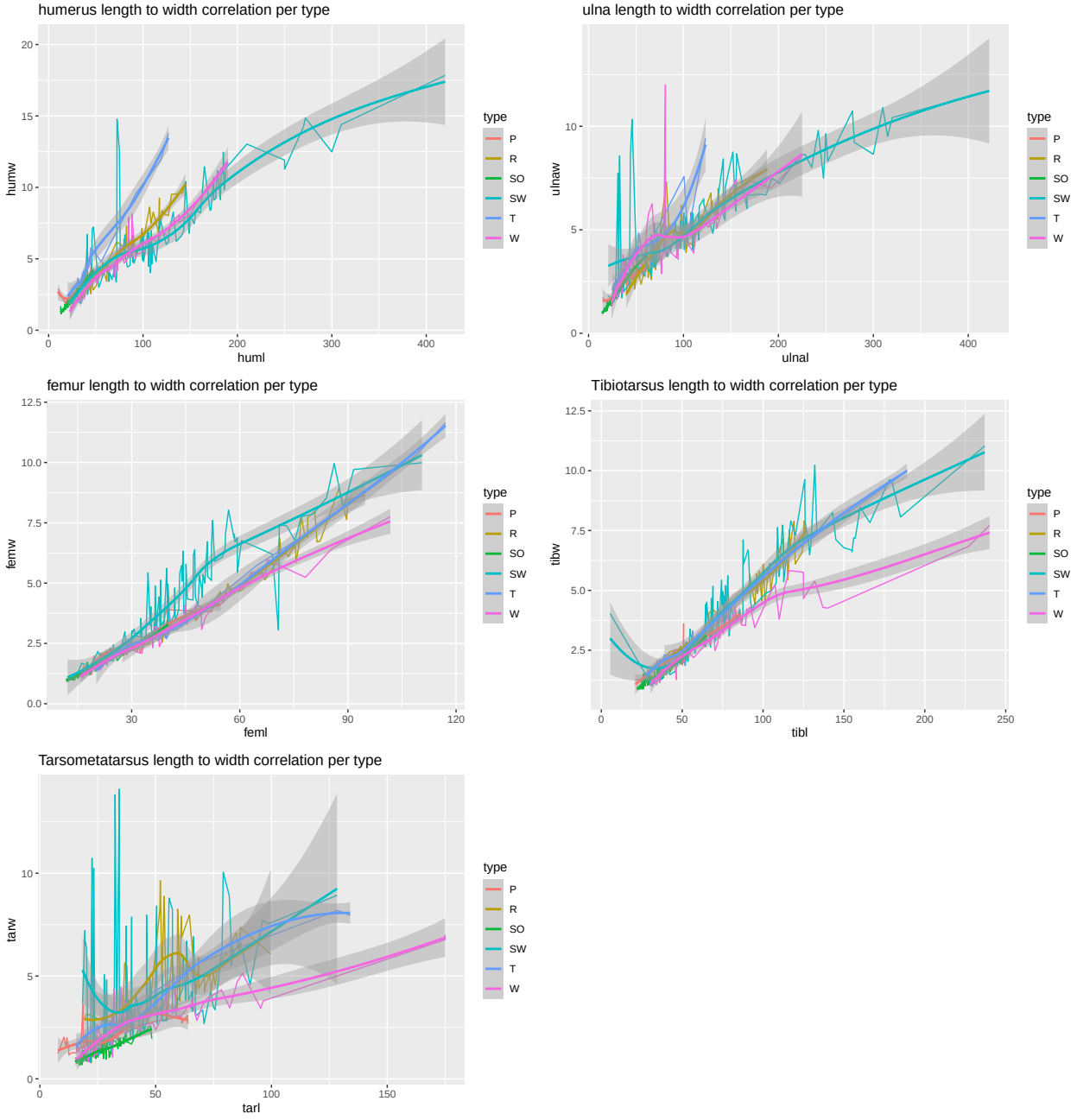


Figure 3: scatterplots for each bone, plotting the length against width, showing the correlation between bird types.

The scatterplots in figure 3 suggest that there is a high correlation between bird types and their bone structures. A correlation matrix was made to see precisely how highly correlated our data is.

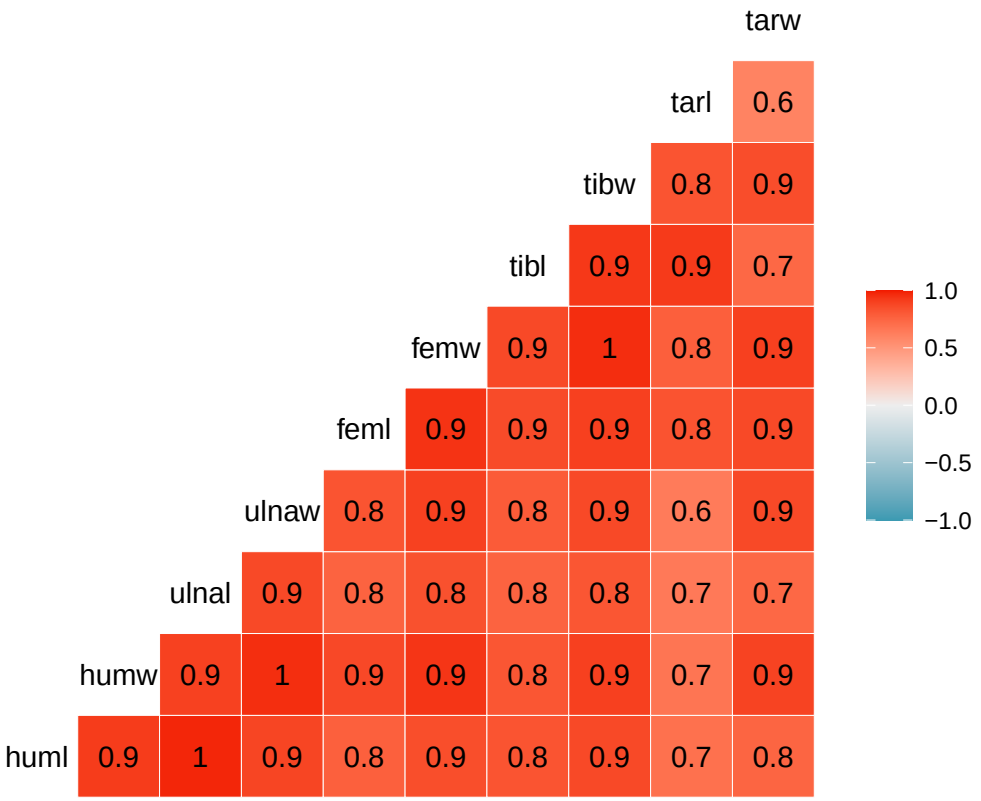


Figure 4: A correlation matrix between the length and width of bones across different bird types.

The matrix in figure 4 shows that all values are above 0 and all of them closer to 1 than 0. The values are thus highly correlated.

4.4 Machine Learning

4.4.1 Algorithm selection

A selection of machine learning models has been made to cover to include representatives of all classifier categories. These include; ZeroR, oneR, J48, Naive Bayes, SMO, K-Nearest Neighbor (IBK), Simple Logistic and Random forest. The algorithms were compared using their default values in weka, using 10-fold cross validation with 100 repetitions.

Table 4: Quality metrics for each algorithm with default values using 100 fold crossvalidation (continued below)

algorithm	percentage correct
weka.classifiers.lazy.IBk	89.16
weka.classifiers.functions.SimpleLogistic	85.24
weka.classifiers.trees.RandomForest	84.54
weka.classifiers.trees.J48	75.26
weka.classifiers.functions.SMO	59.89
weka.classifiers.rules.OneR	55.7
weka.classifiers.bayes.NaiveBayes	50.95
weka.classifiers.rules.ZeroR	30.03

Table 5: Table continues below

False_positive_rate	True_positive_rate	True_negative_rate
0.02258	0.8961	0.9774
0.04788	0.8808	0.9521
0.06709	0.8675	0.9329
0.09467	0.7725	0.9053
0.219	0.8609	0.781
0.1646	0.6763	0.8354
0.03032	0.3105	0.9697
0	0	1

False_negative_rate	F_measure
0.1039	0.9157
0.1192	0.8783
0.1325	0.85
0.2275	0.7643
0.1391	0.7124
0.3237	0.6382
0.6895	0.4401
1	NA

Table **??** shows that each algorithm performs better than the baseline algorithms; ZeroR and OneR. The F-measure of K-nearest neighbor (91.57%), SimpleLogistic (87,83%) and RandomForest (85,00%) score the top 3 highest values.

The top 3 algorithms had their hyperparameters adjusted. The K-nearest neighbor algorithm had the amount of neighbors set from 1 to 3 in an attempt to prevent overfitting. The RandomForest algorithm had their depth set to 10 to prevent overfitting. Finally, the Simplelogistic algorithm couldn't have its parameters

adjusted because the optimal number of LogitBoost iterations to perform is cross-validated, which leads to automatic attribute selection.

Table 7: Quality metrics for the top 3 algorithms with adjusted hyperparameters, using 10-fold crossvalidation with 100 repetitions. (continued below)

algorithm	percentage correct
weka.classifiers.functions.SimpleLogistic	85.08
weka.classifiers.trees.RandomForest	84.73
weka.classifiers.lazy.IBk	82.06

Table 8: Table continues below

False_positive_rate	True_positive_rate	True_negative_rate
0.04983	0.882	0.9502
0.068	0.8645	0.932
0.07367	0.823	0.9263

False_negative_rate	F_measure
0.118	NA
0.1355	NA
0.177	NA

Table **??** shows that the only algorithm that does not reduce its quality metrics is SimpleLogistic. The other algorithms in an attempt to prevent overfitting, have reduced values for all quality metrics. Therefore the SimpleLogistic algorithm is the best contender for the final model.

4.4.2 Attribute selection

In this section the attributes will be selected for their highest information value. The ANOVA test already showed that the humerus and ulna bones are most informative. Using the Weka Explorer, the best attributes were found using attribute evaluators.

Accuracy	**Attribute**
54.9637	1 huml
53.0266	4 ulnaw
52.7845	2 humw
51.3317	3 ulnal
49.6368	7 tibl
47.6998	10 tarw
47.6998	5 feml
46.4891	8 tibw
44.7942	6 femw
42.8571	9 tarl

Table 10: Attribute Selection using the OneR Feature Evaluator + Ranker Method

In figure **??** For the one OneR feature evaluator + ranker method, huml and ulnaw are the highest scoring attributes. This ranking corresponds with the ANOVA test that was performed. The removal of certain

attributes will now be tested on the top 3 algortihms too see if there is an improvement in the model. Multiple rankings were made, to see them please refer to the EDA.

Table 11: Quality metrics for each algorithm, adjusted for removel of lowest attributes (continued below)

algorithm	percentage correct
weka.classifiers.trees.RandomForest	76.51
weka.classifiers.functions.SimpleLogistic	76.36
weka.classifiers.lazy.IBk	74.4

Table 12: Table continues below

False_positive_rate	True_positive_rate	True_negative_rate
0.0825	0.8064	0.9175
0.1044	0.873	0.8956
0.1069	0.8013	0.8931

False_negative_rate	F_measure
0.1936	0.7975
0.127	0.8252
0.1987	0.7747

As can be seen in figure **??**, after removing the lowest attribute all relevant parameters have a worse score. All attributes have contributed to better functioning of the model, therefore none have been removed in the selection of the final model.

4.4.3 ROC curve

This section will visualize the Receiver Operating Characteristics (ROC) curves for the RandomForest, K-nearest neighbor and SimpleLogistic algorithms. ROC curve is a graph showing the performance of a classification model at all threshold settings. This is done by plotting the True Positive Rate against the False Positive Rate, where True Positive Rate is on the y-axis and False Positive rate is on the x-axis.

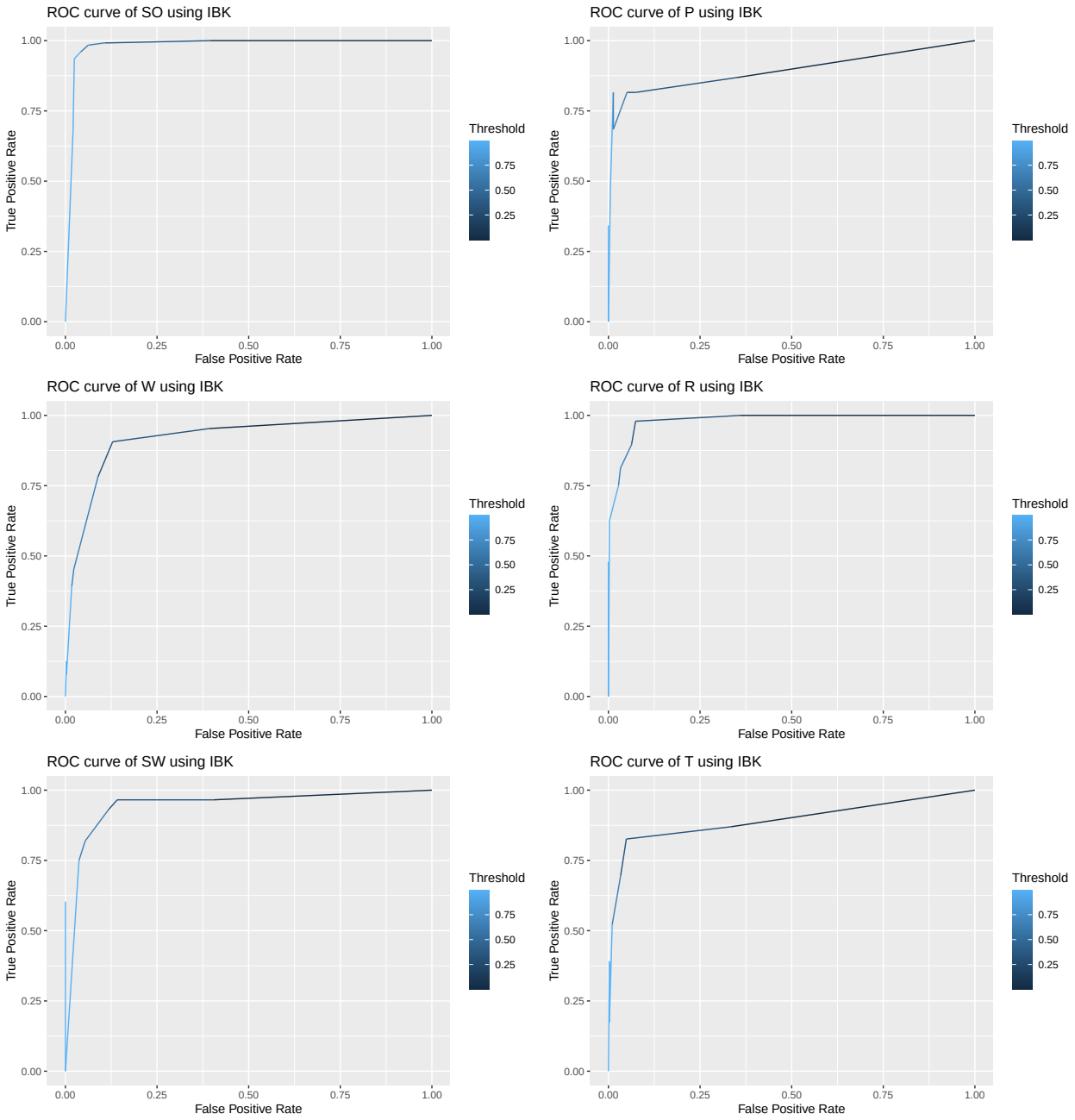


Figure 5: ROC curve of all bird classes using the IBK (K-nearest neighbor) algorithm

Figure **??** shows the ROC curves for all different bird groups. The threshold seems to stop at 0.75 for all of the groups.

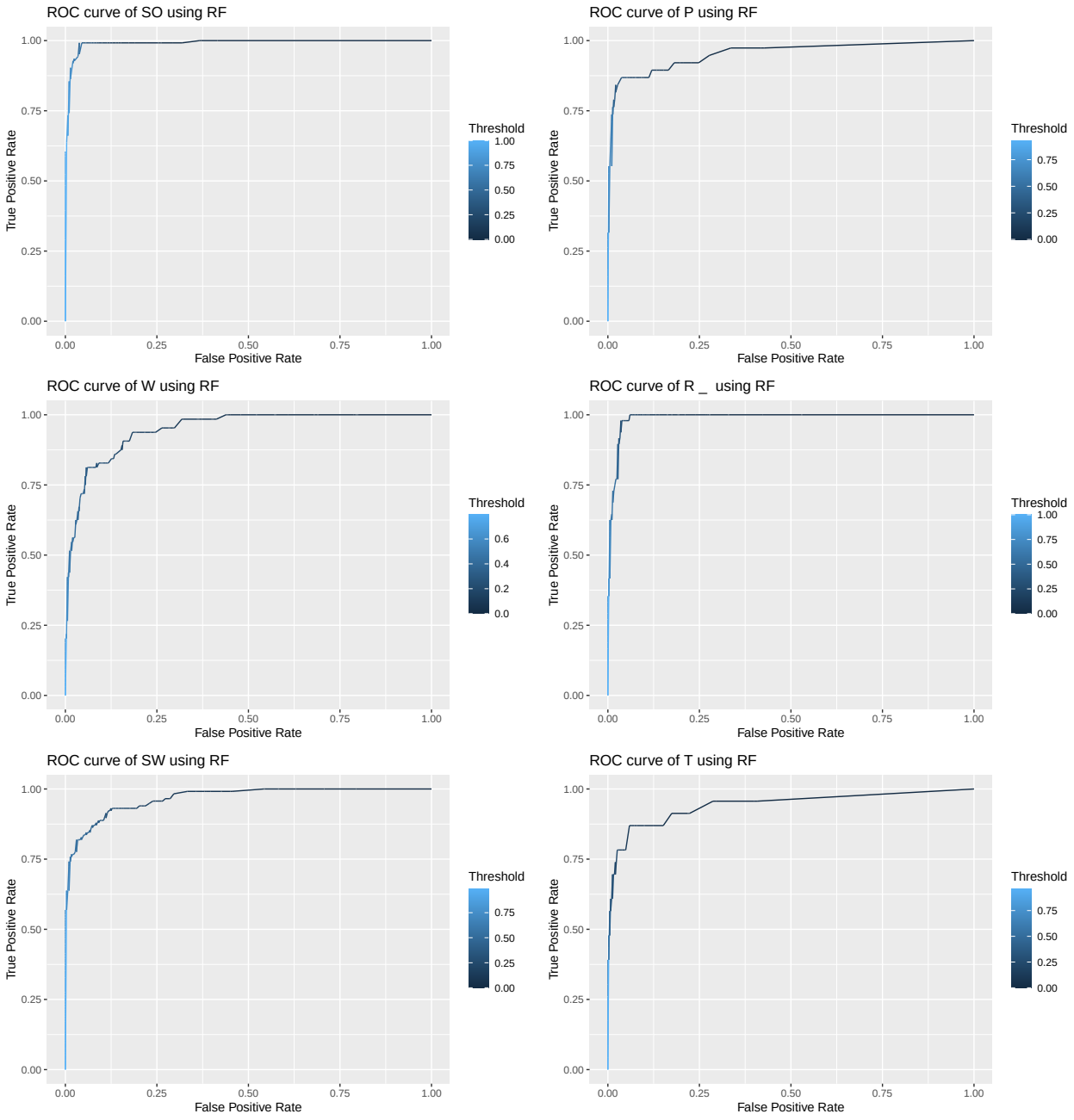


Figure 6: ROC curve of all bird classes using the RF (Random Forest) algorithm, plotting True positives against False positives.

Figure **??** shows the performance of the RandomForest algorithm is an improvement over the curves over the IBK algorithm, with less False positives overall.

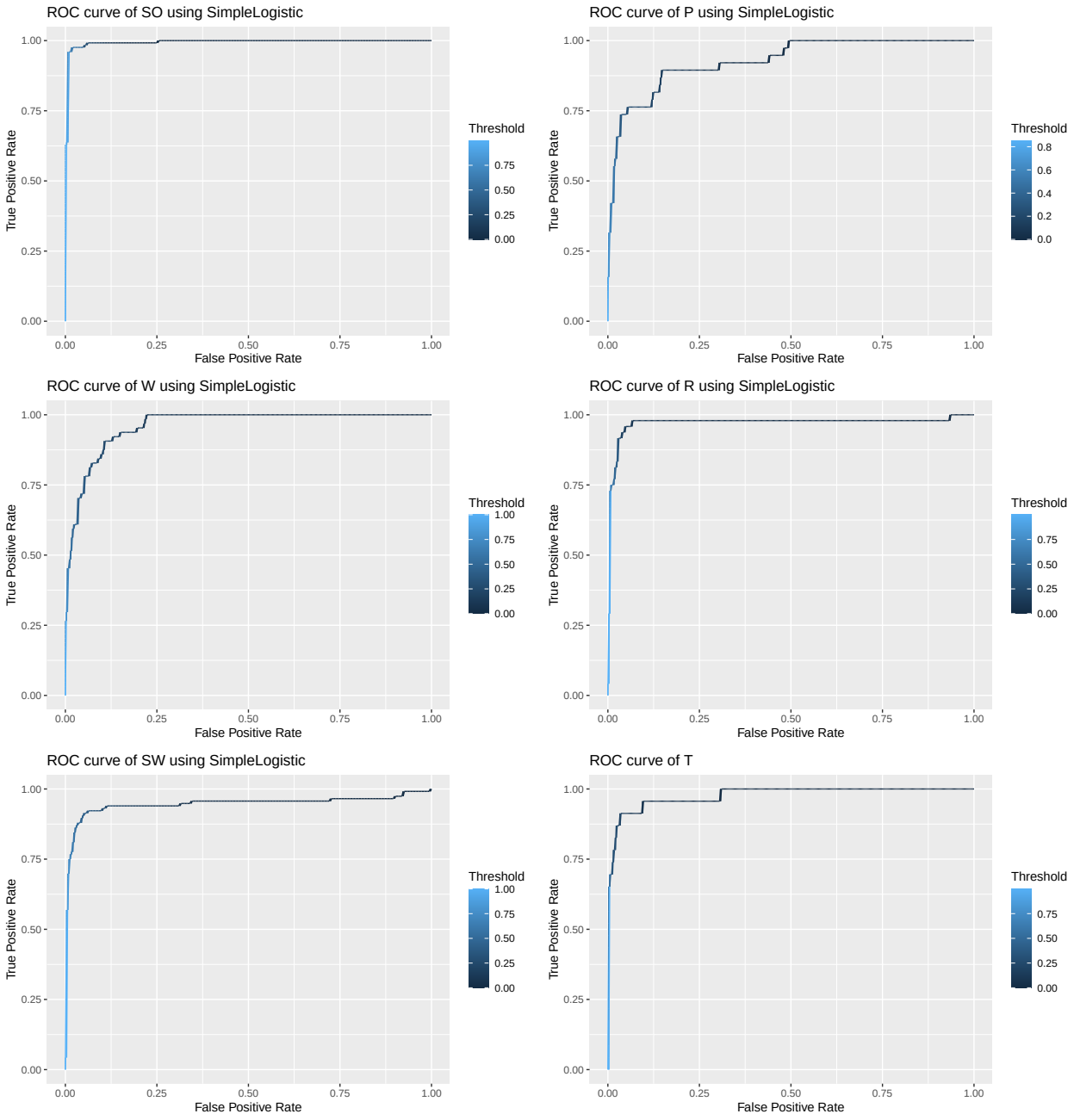


Figure 7: ROC curve of all bird classes using the SL (Simple Logistic) algorithm, plotting True positives against False positives.

Finally the ROC curves of the SimpleLogistic algorithm in figure **??** shows an improved relation between the true and false positives compared to the KNN algorithm. The SimpleLogistic algorithm is on par with the RandomForest algorithm. In Weka, the SimpleLogistic has the AUC value of 0.898, and the AUC value of RandomForest is 0.880. This indicates that the ROC curve of SimpleLogistic is better than the ROC curve of RandomForest. For this reason the SimpleLogistic algorithm will be used in the final model.

5 Discussion

The selected dataset contains 420 bird specimens and consists of 11 attributes. Each bird in the dataset is represented by 10 measurements related to bone lengths and widths, alongside a categorical attribute known as "type," used for classifying birds into ecological groups. The EDA shows it is a 420x10 size continuous values unbalanced multi-class dataset, containing 2% missing values. Figure **??** shows the unequal distribution of the dataset. Over half of the data is attributed to songbirds (SO) with 30.5% and swimming birds (SW) with 27.6%. With terrestrial birds (T) and scansorial birds (P) together making up only 14.5% of the data. Given their random distribution and negligible impact, the decision was made to remove entries with missing values entirely. The logtransformed boxplot in figure **??** showed a reduction in ourliers for the bigger class attributes like SO and SW. Figure **??** shows which measurement will be most informative for machine learning, thereby helping to answer the research question. The p-values of huml (pval: 1.023e-49) and ulnaw (pval:1.244e-48) are the lowest and are thus most informative. The scatterplots in figure 3 suggest that there is a high correlation between bird types and their bone structures. The confusionmatrix in figure 4 shows exactly how correlated the data is. All values are >6 with most of them being closer to 1. The scatterplots highlighted notable variations across different bird types. The correlation analysis further supported our observations, revealing patterns of association between bone dimensions. Which makes sense, since all birds have similar shapes regardless of size. The values are thus highly correlated. The initial data analysis did not suggest a lot of potential for utilizing bird bone measurements as informative features for classifying birds into ecological groups. The presence of strong positive correlations among bone measurements highlights an issue for most classification algorithms.

A selection of machine learning models has been made to cover to include representatives of all classifier categories. These include; ZeroR, oneR, J48, Naive Bayes, SMO, K-Nearest Neighbor (IBK), Simple Logistic and Random forest. The algorithms were compared using their default values in weka, using 10-fold cross validation with 100 repetitions. During algorithm selection the algorithms such as SimpleLogistic, K-nearest neighbor and RandomForest managed to all score above 80% on accuracy and F1-score. Table **??** shows that each algorithm performs better than the baseline algorithms; ZeroR and OneR. The F-measure of K-nearest neighbor (91.57%), SimpleLogistic (87,83%) and RandomForest (85,00%) score the top 3 highest values. The F-scores are very close to 1, this means that there is a decent balance between precision and recall. Which means that despite the quality of the dataset, the algorithms performed very well with default settings. The top 3 algorithms, K-nearest neighbor, RandomForest and SimpleLogistic had their hyperparameters adjusted. The K-nearest neighbor algorithm had the amount of neighbors set from 1 to 3 in an attempt to prevent overfitting. The RandomForest algorithm had their depth set to 10 to prevent overfitting. Finally, the Simplelogistic algorithm couldn't have its parameters adjusted because the optimal number of LogitBoost iterations to perform is cross-validated, which leads to automatic attribute selection. Table **??** shows that the only algorithm that does not reduce its quality metrics is SimpleLogistic. The other algorithms, in an attempt to prevent overfitting, have reduced values for all quality metrics. Dispite that, SimpleLogistic and RandomForest seem to be tied closely.

Attribute selection had been done for each bones length and width. The ANOVA test already showed that the humerus and ulna bones are most informative. Using the Weka Explorer, the best attributes were found using attribute evaluators. After removing the lowest attribute, all relevant quality markers have a worse score. All attributes have contributed to better functioning of the model, therefore none have been removed in the selection of the final model.

The (ROC) curves for the RandomForest, K-nearest neighbor and SimpleLogistic algorithms were visualised to see the performance of a classification model at all threshold settings. With the exception of K-nearest neighbor **??**, the other ROC curves show similarities. Figure **??** shows the performance of the RandomForest algorithm is an improvement over the curves over the IBK algorithm, with less False positives overall. The ROC curves of the SimpleLogistic algorithm in figure **??** shows an improved relation between the true and false positives compared to the KNN algorithm. Though, the values for false positives and false negatives are not critical information for this type of dataset.

The SimpleLogistic algorithm is on par with the RandomForest algorithm. In Weka, the SimpleLogistic has the AUC value of 0.898, and the AUC value of RandomForest is 0.880. This indicates that the ROC curve of

SimpleLogistic is better than the ROC curve of RandomForest. For this reason, and the fact that our F1 score for SimpleLogistic is slightly higher than the RandomForest algorithm, it will be used in the final model. A Java application has been built to make the application of the SimpleLogistic model user-friendly [6].

6 Conclusion

The primary research question wants to adress the informativeness of bone measurements for classifying birds into ecological groups using machine learning algorithms. Exploratory data analysis gave us an examination of the class distribution within the dataset, revealing an unequal distribution of bird species among ecological groups. There was a high positive correlation between bone measurements, finding a strong positive correlation between different bone lengths and diameters per type of bird. Variations of the data were explored, with outliers identified but retained in the dataset because it aids in better classification in the later stages because of the high correltation. Also because the outliers were not significant; bigger birds have bigger bones, and sizes even between bird types vary. The best algorithm turned out to be SimpleLogistic because of its superior ROC curve and F1 values. However, implementation of the RandomForest algorithm would not be out of the question for a follow-up study.

There were some key findings in this study: Firsy, Certain bone measurements demonstrated significant variations across bird types, suggesting their potential utility for species classification. Secondly, ANOVA tests and machine learning model rankings highlighted specific attributes that significantly contribute to classification accuracy, these ones being humerus length and ulna width. Lastly, machine learning models, particularly Simple Logistic and Random Forest exhibited promising results in predicting bird types based on bone measurements.

With this the research question; can be answered. Using machine learning algorithms, which bone measurements are most informative for classifying birds into ecological groups? There is a significant relationship between bird bone measurements and their ecological groups, and bone measurements are informative for accurate classification. The humerus and the ulna bones being most informative, however all bones proved invaluable for creating an effective model. Which concludes that all bones are informative for classifying birds into ecological groups.

Despite the promising results, this study has some limitations. The dataset's distribution and size may influence the generalizability of the findings, making the application of the model to other datasets harder. Future research could involve larger and more diverse datasets to enhance the robustness of the classification model. More data would ensure that sampling training data from the unbalanced class attributes would not lead to overfitting the model. Additionally, incorporating additional morphological features besides bone measurement or exploring advanced machine learning techniques like boosting may further improve the accuracy of classification.

References

[1] Bird bones and habitats: *Kaggle bird dataset*, "Birds bones and Living Habitats" data set, Retrieved from https://www.kaggle.com/datasets/zhangjuefei/birds-bones-and-living-habits on 14-09-2023

[2] RStudio: *Download RStudio*, Download the RStudio IDE, Retrieved from https://www.rstudio.com/products/rstudio/download/ on 10-11-2023

[3] Weka: *Download Weka*, Downloading and installing Weka, Retrieved from https://waikato.github.io/weka-wiki/ then Download, loading and installing Weka on 10-11-2023

[4] Java, Oracle: *Download Java*, Download Java, Retrieved from https://www.java.com/nl/download/ on 10-11-2023

[5] IntelliJ IDEA: *Download IntelliJ*, Download IntelliJ IDEA, Retrieved from https://www.jetbrains.com/idea/download/ on 10-11-2023

[6] Github Repository: *Download Github*, Get information on this project, Retrieved from https://github.com/ishofstede/thema09 on 10-11-2023

Appendix

```
knitr::opts_chunk$set(echo = FALSE)

knitr::opts_chunk$set(cache = TRUE)
#check if package is installed
if (!require(haven)){
  #if not, install it
  install.packages("haven", dependencies = TRUE)
  #if installed, load library
  library(haven)
}
if (!require(tidyverse)){
  install.packages("tidyverse", dependencies = TRUE)
  library(tidyverse)
}
if (!require(dplyr)){
  install.packages("dplyr", dependencies = TRUE)
  library(dplyr)
}
if (!require(pander)){
  install.packages("pander", dependencies = TRUE)
  library(pander)
}
if (!require(summarytools)){
  install.packages("summarytools", dependencies = TRUE)
  library(summarytools)
}
if (!require(ggplot2)){
  install.packages("ggplot2", dependencies = TRUE)
  library(ggplot2)
}
if (!require(GGally)){
  install.packages("GGally", dependencies = TRUE)
  library(GGally)
}
if (!require(gridExtra)){
  install.packages("gridExtra", dependencies = TRUE)
  library(gridExtra)
}

if (!require(RWeka)){
  install.packages("RWeka", dependencies = TRUE)
  library(RWeka)
}
#load the data
bird_data <- read.table('data/bird.csv', sep=",", header = 1)
#create statistics of bird dataset
bird_data %>%
  select (huml, humw, ulnal, ulnaw, feml, femw, tibl, tibw, tarl,
          tarw, type) -> overview
print(dfSummary(overview, graph.magnif = .75,
                method = "render",
```

```
                plain.ascii= FALSE,
                style = 'grid',
                varnumbers = FALSE,
                valid.col = FALSE,
                graph = FALSE))
#remove the ID's and NA's
clean_bird_data <- na.omit(bird_data)
clean_bird_data <- subset(clean_bird_data, select = -id)
#check if any missing values are left
#colSums(is.na(clean_bird_data))
#write the csv file with cleaned dataset.
write.csv(clean_bird_data, "bird_clean.csv", row.names = F)
#create a barchart per species
ggplot(clean_bird_data, aes(x = type)) + geom_bar(fill = "black") +
  labs(title = "distribution of bird species", x = "species", y = "count")
#create a long format of the data for ggplot2
long_bird_data <- gather(clean_bird_data, key = "Bone", value = "Measurement",
                         huml, humw, ulnal, ulnaw, feml, femw, tibl, tibw,
                        tarl, tarw)

#create a boxplot for each bone measurement
ggplot(long_bird_data, aes(x = type, y = Measurement)) +
  geom_boxplot() +
  facet_wrap(~ Bone, scales = "free_y", nrow = 2) +
  labs(title = "Bone measurement per type of bird in mm")
#create a boxplot for each bone measurement
ggplot(long_bird_data, aes(x = type, y = Measurement)) +
  geom_boxplot() + scale_y_continuous(trans = "log10") +
  facet_wrap(~ Bone, scales = "free_y", nrow = 2) +
  labs(title = "Logtransformed bone measurement per type of bird in mm")

#create a long format of the data for ggplot2
long_bird_data <- gather(clean_bird_data, key = "Bone", value = "Measurement",
                         huml, humw, ulnal, ulnaw, feml, femw, tibl, tibw,
                         tarl, tarw)

#perform ANOVA for each bone measurement
anova_results <- long_bird_data %>%
  group_by(Bone) %>%
  reframe(p_value = anova(aov(Measurement ~ type))$`Pr(>F)`)

#sort the p-values to determine informativeness
anova_results <- na.omit(anova_results)
sorted_anova_results <- anova_results %>%
  arrange(p_value)

#Make table of p-values
pander(sorted_anova_results, caption = "missing values per type")
#create a scatterplot of the bone lengths and widths.
ggplot(clean_bird_data,aes(x=huml,y=humw,color=type))+geom_line()+
  geom_smooth() + labs(title="humerus length to width correlation per type")
ggplot(clean_bird_data,aes(x=ulnal,y=ulnaw,color=type))+geom_line()+
  geom_smooth() + labs(title="ulna length to width correlation per type")
```

```r
ggplot(clean_bird_data,aes(x=feml,y=femw,color=type))+geom_line()+
  geom_smooth() + labs(title="femur length to width correlation per type")
ggplot(clean_bird_data,aes(x=tibl,y=tibw,color=type))+geom_line()+
  geom_smooth() + labs(title="Tibiotarsus length to width correlation per type")
ggplot(clean_bird_data,aes(x=tarl,y=tarw,color=type))+geom_line()+
  geom_smooth() +
  labs(title="Tarsometatarsus length to width correlation per type")
ggcorr(clean_bird_data, label = TRUE)
# Load results into R
weka_default <- read.csv("data/weka_exp_default.csv", header=T)
results_default <- aggregate(weka_default[c("Percent_correct",
                                            "False_positive_rate",
                                            "True_positive_rate",
                                            "True_negative_rate",
                                            "False_negative_rate",
                                            "F_measure")],
                             list(weka_default$Key_Scheme), mean)
colnames(results_default) <- c("algorithm", "percentage correct",
                               "False_positive_rate",
                               "True_positive_rate",
                               "True_negative_rate",
                               "False_negative_rate",
                               "F_measure")
# Order the results
results_default <- results_default %>%
  arrange(desc(`percentage correct`))
pander(results_default,
       caption="Quality metrics for each algorithm with default values using 100 fold crossvalidation")
#load results into R
weka_default <- read.csv("data/weka_exp_forest_knn_simple.csv", header=T)
results_default <- aggregate(weka_default[c("Percent_correct",
                                            "False_positive_rate",
                                            "True_positive_rate",
                                            "True_negative_rate",
                                            "False_negative_rate",
                                            "F_measure")],
                             list(weka_default$Key_Scheme), mean)
colnames(results_default) <- c("algorithm",
                               "percentage correct",
                               "False_positive_rate",
                               "True_positive_rate",
                               "True_negative_rate",
                               "False_negative_rate",
                               "F_measure")
#order the results
results_default <- results_default %>%
  arrange(desc(`percentage correct`))
pander(results_default,
       caption="Quality metrics for the top 3 algorithms with adjusted hyperparameters, using 10-fold c
#Load results into R
weka_default <- read.csv("data/weka_exp_datasets.csv", header=T)
results_default <- aggregate(weka_default[c("Percent_correct",
                                            "False_positive_rate",
```

```
                                          "True_positive_rate",
                                          "True_negative_rate",
                                          "False_negative_rate",
                                          "F_measure")],
                              list(weka_default$Key_Scheme), mean)
colnames(results_default) <- c("algorithm",
                               "percentage correct",
                               "False_positive_rate",
                               "True_positive_rate",
                               "True_negative_rate",
                               "False_negative_rate",
                               "F_measure")
#order the results
results_default <- results_default %>%
  arrange(desc(`percentage correct`))
pander(results_default,
       caption="Quality metrics for each algorithm, adjusted for removel of lowest attributes")
#function to generate ROC curve
SO_curve_IBK <- read.arff("data/SO_curve_IBK.arff")
P_curve_IBK <- read.arff("data/P_curve_IBK.arff")
W_curve_IBK <- read.arff("data/W_curve_IBK.arff")
R_curve_IBK <- read.arff("data/R_curve_IBK.arff")
SW_curve_IBK <- read.arff("data/SW_curve_IBK.arff")
T_curve_IBK <- read.arff("data/T_curve_IBK.arff")
ggplot(SO_curve_IBK, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                     colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of SO using IBK")
ggplot(P_curve_IBK, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                     colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of P using IBK")
ggplot(W_curve_IBK, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                     colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of W using IBK")
ggplot(R_curve_IBK, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                     colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of R using IBK")
ggplot(SW_curve_IBK, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                     colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of SW using IBK")
ggplot(T_curve_IBK, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                     colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of T using IBK")
#function to generate ROC curve
SO_curve_RF <- read.arff("data/SO_curve_RF.arff")
P_curve_RF <- read.arff("data/P_curve_RF.arff")
W_curve_RF <- read.arff("data/W_curve_RF.arff")
R_curve_RF <- read.arff("data/R_curve_RF.arff")
SW_curve_RF <- read.arff("data/SW_curve_RF.arff")
T_curve_RF <- read.arff("data/T_curve_RF.arff")
ggplot(SO_curve_RF, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                     colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of SO using RF")
ggplot(P_curve_RF, aes(x=`False Positive Rate`, y=`True Positive Rate`,
```

```
                    colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of P using RF")
ggplot(W_curve_RF, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                    colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of W using RF")
ggplot(R_curve_RF, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                    colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of R _  using RF")
ggplot(SW_curve_RF, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                     colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of SW using RF")
ggplot(T_curve_RF, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                    colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of T using RF")
SO_curve <- read.arff("data/SO_curve.arff")
P_curve <- read.arff("data/P_curve.arff")
W_curve <- read.arff("data/W_curve.arff")
R_curve <- read.arff("data/R_curve.arff")
SW_curve <- read.arff("data/SW_curve.arff")
T_curve <- read.arff("data/T_curve.arff")
ggplot(SO_curve, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                     colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of SO using SimpleLogistic")
ggplot(P_curve, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                    colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of P using SimpleLogistic")
ggplot(W_curve, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                    colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of W using SimpleLogistic")
ggplot(R_curve, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                    colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of R using SimpleLogistic")
ggplot(SW_curve, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                     colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of SW using SimpleLogistic")
ggplot(T_curve, aes(x=`False Positive Rate`, y=`True Positive Rate`,
                    colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of T")
```