thema09_logboek

Isabella Hofstede

2023-09-13

Exploratory Data Analysis

For the purpose of this project a few packages and libraries will be installed. All of these are for the purpose of either plotting or formatting the data or plots. Installing haven for formatting:

```
#check if package is installed
if (!require(haven)){
  #if not, install it
  install.packages("haven", dependencies = TRUE)
  #if installed, load library
  library(haven)
}
```

Installing tidyverse for formatting:

```
if (!require(tidyverse)){
  install.packages("tidyverse", dependencies = TRUE)
  library(tidyverse)
}
```

Installing dply for formatting:

```
if (!require(dplyr)){
  install.packages("dplyr", dependencies = TRUE)
  library(dplyr)
}
```

Installing Pander for formatting:

```
if (!require(pander)){
  install.packages("pander", dependencies = TRUE)
  library(pander)
}
```

Installing summarytools for formatting:

```
if (!require(summarytools)){
  install.packages("summarytools", dependencies = TRUE)
  library(summarytools)
}
```

Installing ggplot2 for plotting:

```
if (!require(ggplot2)){
  install.packages("ggplot2", dependencies = TRUE)
  library(ggplot2)
}
```

Installing GGally for plotting:

```
if (!require(GGally)){
  install.packages("GGally", dependencies = TRUE)
  library(GGally)
}
```

Installing GridExtra for formatting:

```
if (!require(gridExtra)){
  install.packages("gridExtra", dependencies = TRUE)
  library(gridExtra)
}
```

Installing Rweka for plotting ROC curve:

```
if (!require(RWeka)){
  install.packages("RWeka", dependencies = TRUE)
  library(RWeka)
}
```

Context:

A dataset has been chosen with the aim of creating a supervised ML model for classification. We picked a dataset on bird classification by bone structure. There are different types of birds, ranging from flying, to swimming, to running birds. For this dataset, birds have been classified into ecological groups according to their habitats. There are 8 ecological groups of birds:

- Swimming Birds;
- Wading Birds;
- Terrestrial Birds;
- Raptors;
- Scansorial Birds;
- Singing Birds;
- Cursorial Birds (not included in dataset);
- Marine Birds (not included in dataset).

The first 6 groups are covered by this dataset. Birds belonging to different ecological groups have different appearances: flying birds have strong wings and wading birds have long legs. Their living habits are somewhat reflected in their bones' shapes. We may think of examining the underlying relationship between sizes of bones and ecological groups , and recognizing bird's ecological groups by their bone structure.

Status of data

The dataset comes from Kaggle (https://www.kaggle.com/datasets/zhangjuefei/birds-bones-and-living-habits). It is made up of measurements of bird skeletons from collections of the Natural History Museum of Los Angeles County (data provided by Dr. D. Liu of Beijing Museum of Natural History). 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):

- Huml and Humw: Length and Width of Humerus;
- Ulnal and Ulnaw: Length and Width of Ulna;
- Feml and Femw: Length and Width of Femur;
- Tibl and Tilw: Length and Width of Tibiotarsus;
- Tarl and Tarw: Length and Width of Tarsometatarsus.

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:

- SW: Swimming Birds;
- W: Wading Birds;
- T: Terrestrial Birds;
- R: Raptors;
- P: Scansorial Birds;
- SO: Singing Birds.

Loading data

We will load our data into R, making sure to separate by commas.

```
#load the data
bird_data <- read.table('bird.csv', sep=",", header = 1)
```

Univariate analysis

Summary statistics

Firstly, we need to have an overview of our data. Pictured below is a summary of the dataset:

```
#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))
```

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

There are 15 missing values. Let's see which birds those values are attached to;

```
#count missing values per type
missing_data_per_type <- sapply(split(bird_data, bird_data$type),
                                function(subset) {
                                  sum(apply(is.na(subset),
                                            1, any))
                                  })

#make table of missing values
result_af <- data_frame(Type = names(missing_data_per_type),
                        Missing_values= missing_data_per_type)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## i Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
pander(result_af, caption = "missing values per type")
```

Table 1: missing values per type

Type	Missing_values
P	0
R	2
SO	4
SW	0
T	0
W	1

So there are a total of 7 birds with missing values. This means there are a total 420 entries with 15 missing values spread across 7 birds. The missing values are the measurements of certain bones but none of the main attribute data; "type", is missing. Apparently there are a few birds with "missing bones". This means a choice must be made to discard the missing data or perform imputation, which means replacing the missing data with substitutions. However 7 out of 420 means the observations with missing values account for less than 3% of the dataset, and may be insignificant enough to remove entirely. Let's also think about why the data is missing; is it significant to specific types of birds or is the data quite random?

These missing values are most likely field data that was not properly formatted into the csv file. Since the SO (songbirds) are over represented in the data, it seems fine to remove the entries with missing values. The missing values also have no real meaning, since every bird is supposed to have the same amount of bones, which means the missing values are just bones that were not measured.

Cleaning dataset

The missing values and the id column shall be removed and a new CSV file will be created. This will be the clean dataset with which we will perform machine learning.

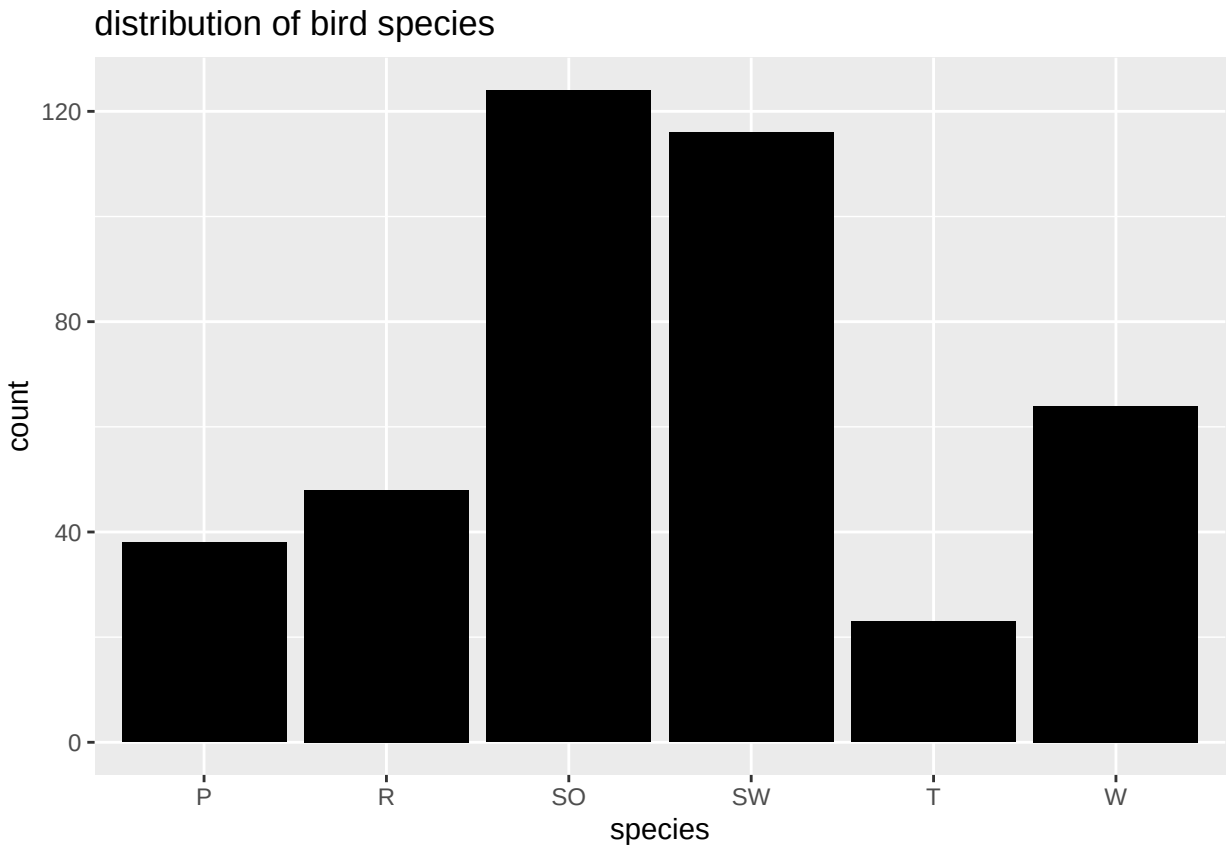
```
#remove the ID's and NA's
clean_bird_data <- na.omit(bird_data)
clean_bird_data <- subset(clean_bird_data, select = -id)
#check if any missing values are left
colSums(is.na(clean_bird_data))
#write the csv file with cleaned dataset.
write.csv(clean_bird_data, "bird_clean.csv", row.names = F)
```

All the missing values have been removed and the cleaned data can now be used for further analysis.

Class distribution

We need to check whether or not the data is equally distributed between the types of birds.

```
#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")
```



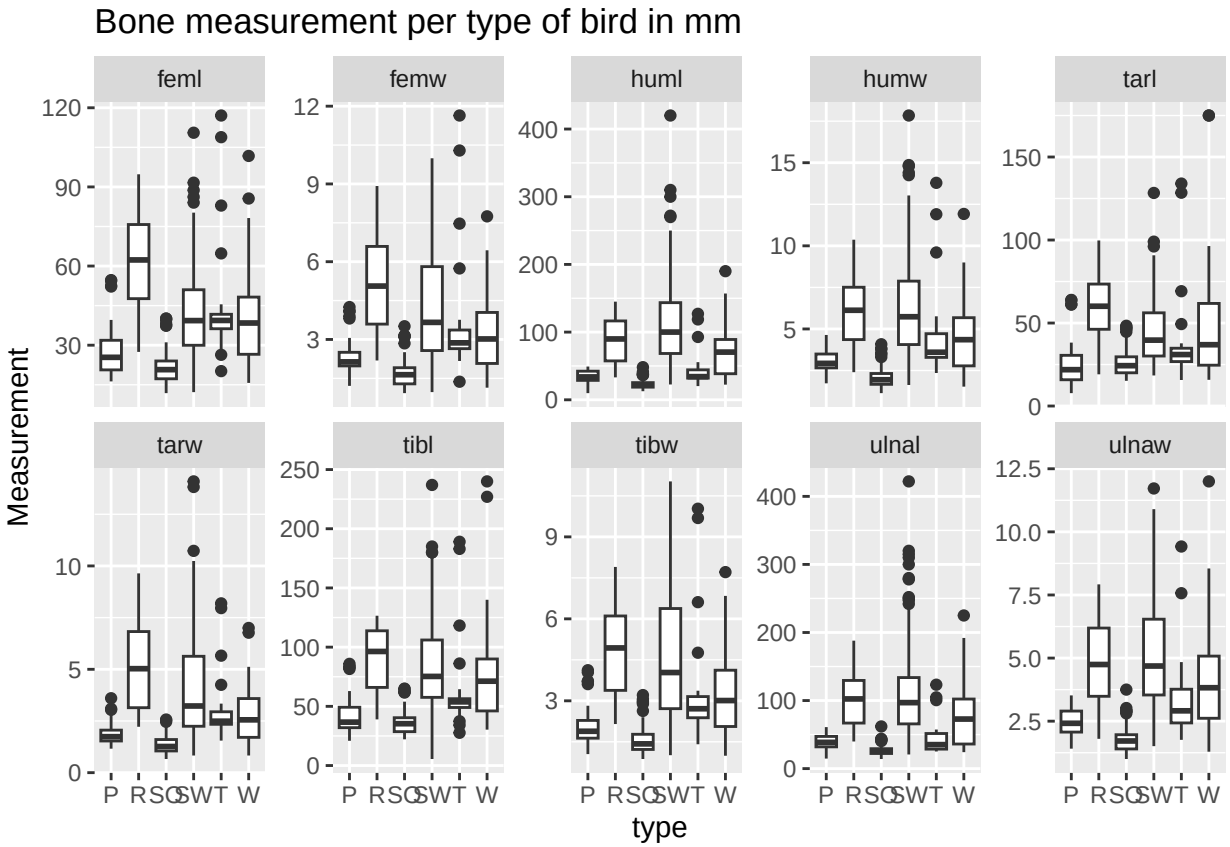
On a glance we can already see that the distribution is not equal, it is seemingly all over the place. There is an over representation of songbirds (SO) and Swimming birds (SW). Terrestrial birds (T) are the group with the least instances. Wading birds (W), Raptos (R) and Scansorial birds (P) are somewhere in between. The 6 classes are not equally distributed. To remedy this, we may double the lowest group or half the biggest group if the training models are insufficiently trained. For now though, we shall use the dataset as is without any further changes.

Bivariate

Variation and distributions

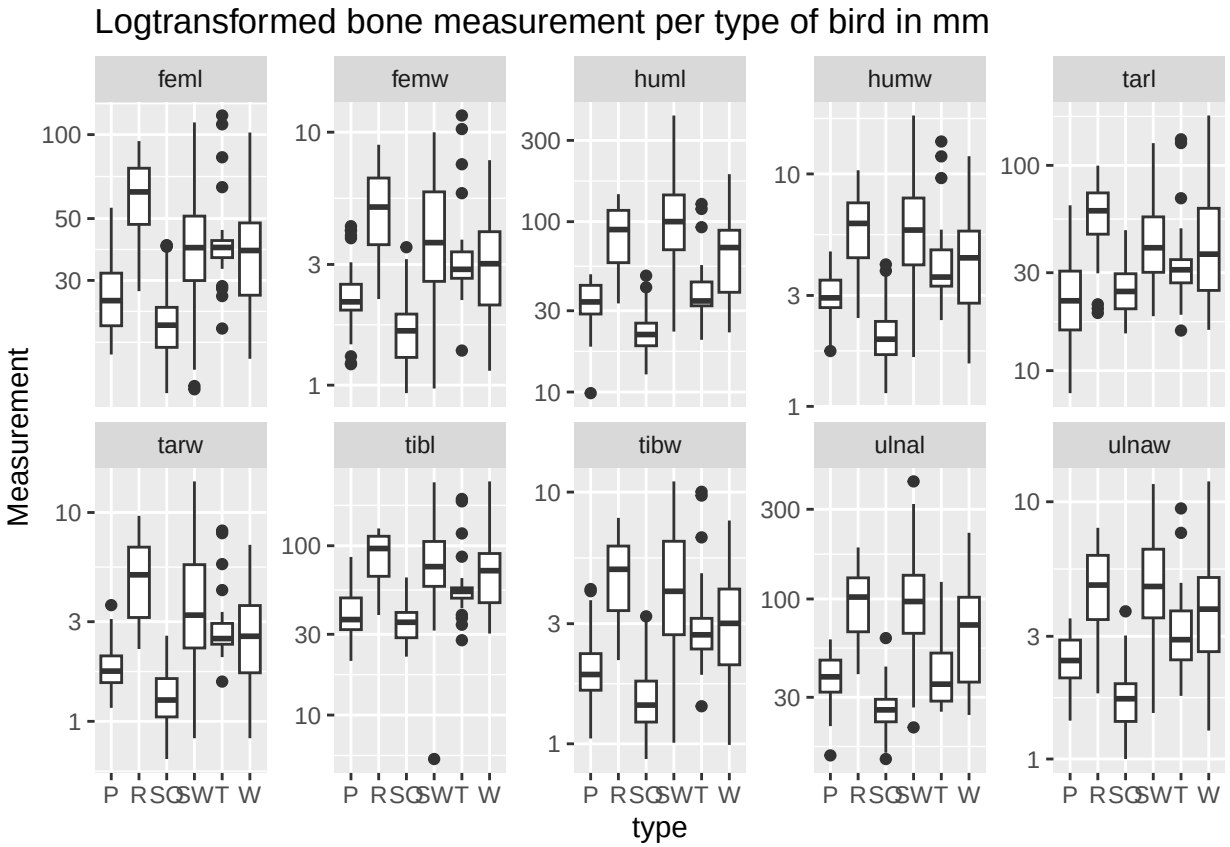
To check the distribution of the data we can make boxplots of bone types for each type of bird. We want to look for outliers in each type.

```
#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")
```



The data seems to have a lot of outliers, to see whether or not this is relevant we can do a log transformation to check the significance. Usually this will not be done

```
#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")
```



Most outliers of the SW group have been removed entirely. There are still plenty of outliers in T group, but since it's the smallest group no instances shall be removed. It is safe to assume that the outliers are actually just bigger and smaller birds. From now on we can assume the outliers are not significant enough to be removed and thus the regular dataset will be used for machine learning. This data can be used for trees, but if these algorithms prove insufficient, logtransformed data may be used for Naive Bayes, depending on how well the other algorithms will do.

Class labels

How can we see if a variable is informative for machine learning? To find out, we will perform an ANOVA on each bone type per bird type and sort the p-values. This will help us determine the most useful values for prediction, even though the distribution is skewed. This will be valuable information for building algorithms with trees.

```
#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)
```

```r
sorted_anova_results <- anova_results %>%
  arrange(p_value)

#Make table of p-values
pander(sorted_anova_results, caption = "missing values per type")
```

Table 2: missing values per type The 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. Even so, almost all values seem to be reasonably close together. When making our mode, we will remove the lowest variables like tarl and tibl to see if it improves the final results.

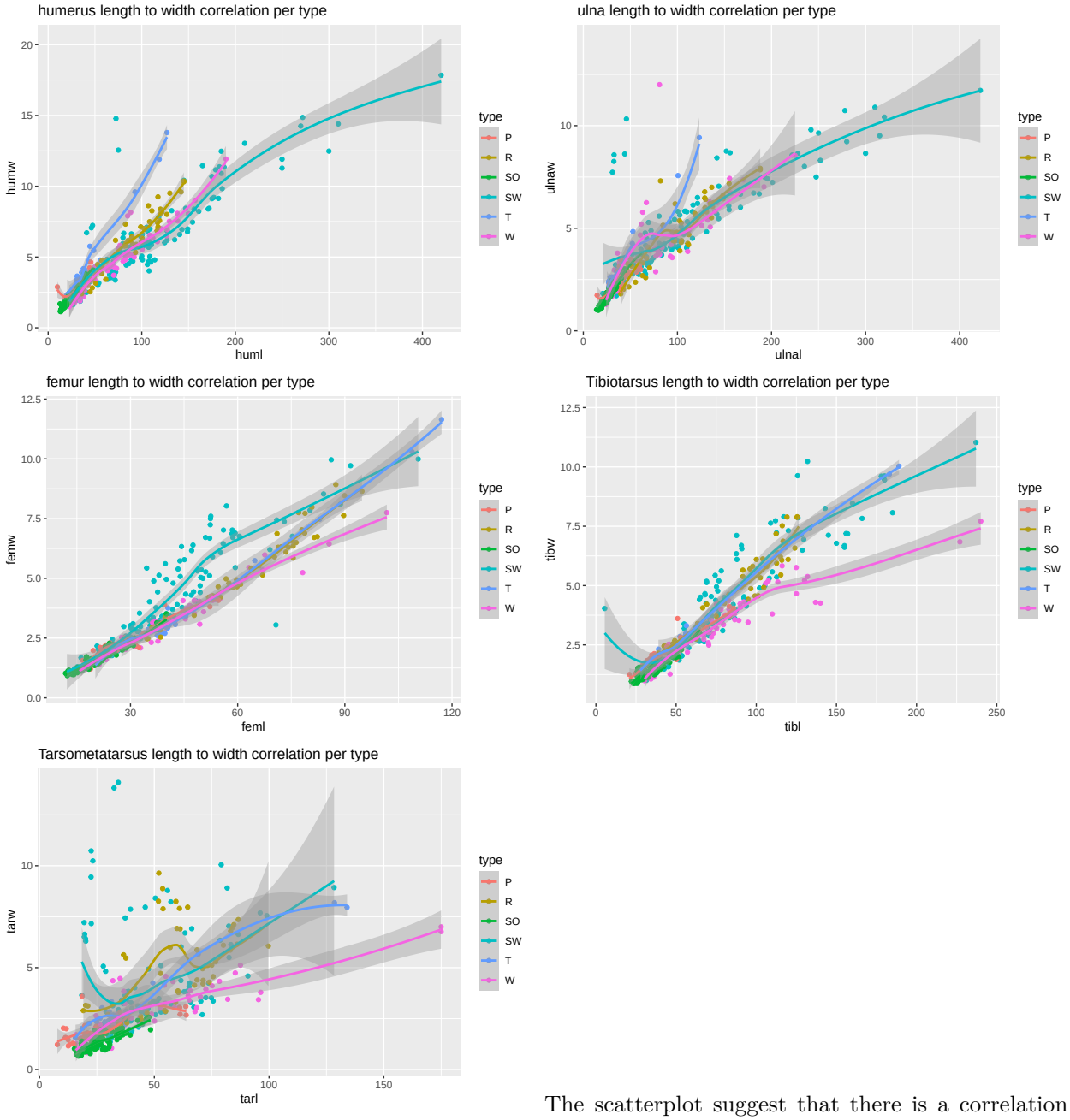
Bone	p_value
huml	1.023e-49
ulnaw	1.244e-48
feml	8.762e-46
tibw	1.785e-44
humw	2.766e-44
femw	6.326e-42
tarw	4.201e-40
ulnal	2.554e-39
tibl	2.366e-36
tarl	3.724e-24

Multivariate

Correlation

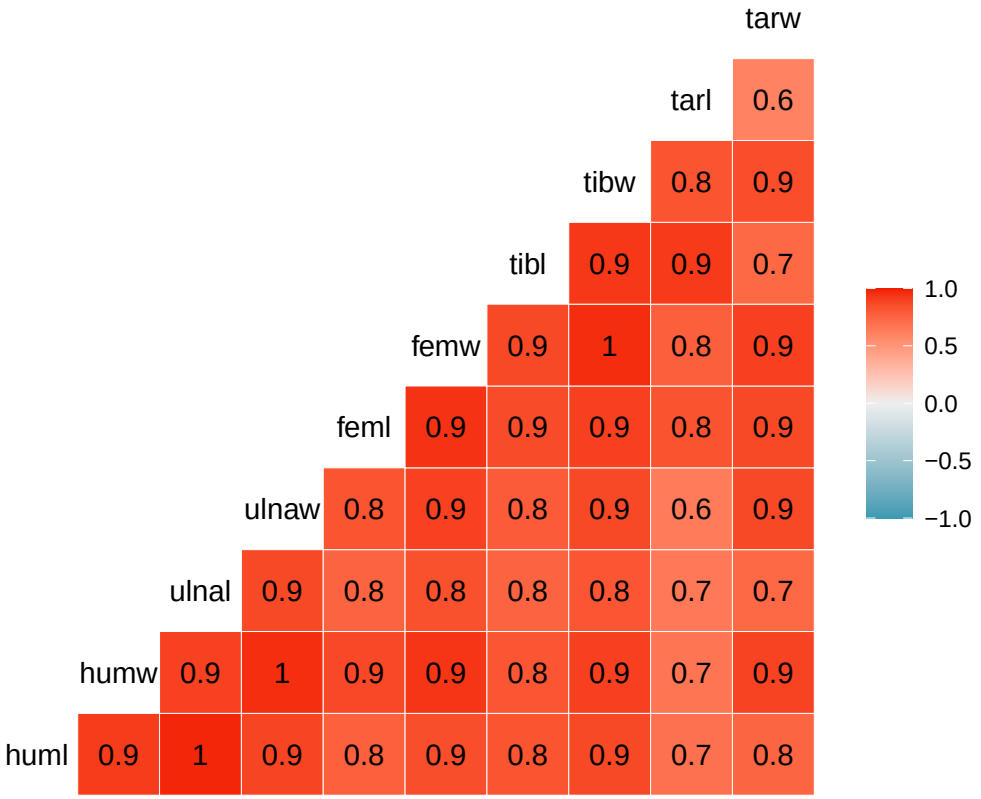
Before checking, the logical assumption would be that the data is highly correlated, since all birds have similar bone structures and the birds within a specific type are going to look even more similar. To be sure, however, we can use a scatterplot to visualize the distribution of the bone sizes and the relationship between them.

```r
#create a scatterplot of the bone lengths and widths.
ggplot(clean_bird_data,aes(x=huml,y=humw,color=type))+geom_point()+
  geom_smooth() + labs(title="humerus length to width correlation per type")
ggplot(clean_bird_data,aes(x=ulnal,y=ulnaw,color=type))+geom_point()+
  geom_smooth() + labs(title="ulna length to width correlation per type")
ggplot(clean_bird_data,aes(x=feml,y=femw,color=type))+geom_point()+
  geom_smooth() + labs(title="femur length to width correlation per type")
ggplot(clean_bird_data,aes(x=tibl,y=tibw,color=type))+geom_point()+
  geom_smooth() + labs(title="Tibiotarsus length to width correlation per type")
ggplot(clean_bird_data,aes(x=tarl,y=tarw,color=type))+geom_point()+
  geom_smooth() +
  labs(title="Tarsometatarsus length to width correlation per type")
```



The scatterplot suggest that there is a correlation between bird types and their bone structures. We can also look at a correlation matrix to see precisely how highly correlated our data is:

```
ggcorr(clean_bird_data, label = TRUE)
```



Unsurprisingly the correlation matrix also gave us a large positive correlation between all our measurements. This is of course because when birds get bigger, so do their bone measurements. The swimming birds (SW), however, seem a lot bigger than the others. It makes sense that SW has a lot of different measurements because big birds, like swans, and small birds, like coots, are both classified as swimming birds. Since the correlation between different bone measurements is so high, classification could prove difficult. We need to keep this in mind when choosing our model.

Research Question

Based on the exploratory data analysis the research question will be: Using machine learning algorithms, which bone measurements are most informative for classifying birds into ecological groups?

Which then follows that the null hypothesis will be: There is no significant relationship between bird bone measurements and their ecological groups, and bone measurements are not informative for accurate classification.

And so the alternative hypothesis: There is a significant relationship between bird bone measurements and their ecological groups, and bone measurements are informative for accurate classification.

Supervised Machine Learning

Quality metrics

To assess the performance of our algorithm there are a few metrics that are commonly used:

- Accuracy/error rate: measures the ratio of correct/incorrect prediction to the total number of predictions, giving us an idea how often the model is correct;

- Precision/recall: measures the accuracy of positive predictions. Important to determine true positives and false positives rate;
- F1 score: mean of precision/recall. Useful for we want to give equal consideration to both true positives and false negatives;
- Specificity: measures true negative rate. Used when true negatives are crucial to know.
- Confusion matrix: used to present a breakdown of true positive, true negative, false positive and false negative counts.
- ROC curve: A curve showing models performance across thresholds. Helps visualize trade-off between true positive rate and false positive rate.
- speed: the speed with which an algorithm can make predictions.

We will use a confusion matrix to showcase our most important metrics. The one metric that seems specifically useful for unbalanced datasets is F1 score, this will be shown separately. Speed is not a metric that is important for this relatively low amount of data.

Machine learning in weka

Considering our dataset is highly imbalanced we can already guess that algorithms that don't rely on balancing, like trees or boosting algorithms, will perform better than those that do. Regardless, the selection of machine learning models has been made to cover to include representatives of all classifier categories:

- ZeroR (baseline measurement)
- oneR (baseline measurement)
- J48
- Naive Bayes
- SMO
- K-Nearest Neighbor (IBK)
- Simple Logistic Regression
- Random forest

First we will compare all the algorithms with their default values in weka using the clean dataset in the weka experimenter. For settings, we use 10-fold cross validation with 100 repetitions.

```
# Load results into R
weka_default <- read.csv("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")
```

Table 3: Quality metrics for each algorithm with default values (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 4: 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

All the chosen algorithms scored higher than ZeroR and OneR on most metrics. The most promising results come from the K-Nearest Neighbor (89.16% correct), Simple logistic (85.24% correct) and the Random Forest (84.54% correct) algorithm. The false positives and negatives are also very low in the top 3 algorithms. The F-scores are very close to 1, this means that there is a decent balance between precision and recall. the hyper parameters shall be adjusted to see if the accuracy can be heightened.

KNN scoring above RandomForest is suspicious, especially since the data is highly correlated. This means KNN might be overfitted on default values (KNN=1), so we shall adjust that by using k = log(number of samples) meaning log(4620) = 3.66 -> k = 4 but because KNN is inherently overfitted we can use a slightly lower threshold, we choose knn = 3. As for SimpleLogistic, the optimal number of LogitBoost iterations to perform is cross-validated, which leads to automatic attribute selection. Because of this we shall not adjust the parameters for LogitBoost All other algorithms below 80% will be removed.

- Use 100 fold crossvalidation
- Adjustments KNN: set KNN to 3 instead of 1.

- Adjustments SimpleLogistic: linear regression was used as calibrator.
- Adjustments RandomForest: set depth to 10.

```
# Load results into R
weka_default <- read.csv("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 each algorithm with default values")
```

Table 6: Quality metrics for each algorithm with default values (continued below)

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

Table 7: 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

Using the experimenter we determined that increasing neighbors in KNN decreased accuracy, an indication of overfitting. RandomForest did not seem to increase or decrease in value from increasing the depth of the tree. SimpleLogistic's quality metrics have not increased nor decreased significantly and seems to be the perfect best fitting model for our dataset with nearly 85% accuracy. Next we will check the ROC curve of the SimpleLogistic algorithm to visualize our metrics.

Attribute selection

Loading in the cleaned dataset, we use the weka Explorer to select attributes. We mix and match different search methods with attribute evaluators to find the commonly highest ranking attributes, these will be used in our final model.

=== 1. OneR evaluator + Attribute Ranking settings ===

- Using clean_bird_data as input.
- Search Method: Attribute ranking.
- Attribute Evaluator (supervised, Class (nominal): 11 type): OneR feature evaluator.
- Using 10 fold cross validation for evaluating attributes.
- Minimum bucket size for OneR: 6

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

For the one OneR feature evaluator + ranker method it seems huml and ulnaw is the highest scoring attribute. This ranking seems to correspond with our ANOVA test.

=== 2. CFS subset Evaluator + Exhaustive Search settings ===

- Search Method: Exhaustive Search.
- Start set: no attributes
- Number of evaluations: 1024
- Merit of best subset found: 0.427
- Attribute Subset Evaluator (supervised, Class (nominal): 11 type):
- CFS Subset Evaluator
- Including locally predictive attributes

Attribute
huml
ulnal
ulnaw
tarw

- Selected attributes: 1,3,4,10 : 4

For CFS subset evaluator + exhaustive search method again it seems huml and unlal are highest scoring.

=== 3. Wrapper Subset Evaluate + Exhaustive Search ===

- Exhaustive Search.
- Start set: no attributes
- Number of evaluations: 1024
- Merit of best subset found: 0.897
- Attribute Subset Evaluator (supervised, Class (nominal): 11 type):
- Wrapper Subset Evaluator
- Learning scheme: weka.classifiers.lazy.IBk
- Scheme options: -K 1 -W 0 -A weka.core.neighboursearch.LinearNNSearch -A "weka.core.EuclideanDistance -R first-last"
- Subset evaluation: classification accuracy
- Number of folds for accuracy estimation: 5
- Selected attributes: 1,2,3,5,6,7,9,10 : 8

Attribute
huml
humw
ulnal
feml
femw
tibl
tarl
tarw

For Wrapper subset evaluate + exhaustive search it seems huml and humw are the highest scoring.

Overall it seems the length and width of the humerus and ulna are the most significant measurements for accurate classification of bird groups. These bones are located in the wings of the bird. It makes sense that wingspan would determine the group that the bird belongs to. Lagging behind in ranking are the femur and tibiotarsus bones, which are located in the legs of the bird. All variables however, lie above the 40% accuracy. Which is certainly not insignificant. We will experiment with removing multiple variables from our models and see if the final model's accuracy will improve.

Inside the weka explorer we removed individual attributes, being sure to keep our class attribute. We then save all these new datasets as .csv files. In the experimenter we check which mix of attributes and algorithms produce the best result. we run 100 repetitions and use 10-fold crossvalidation. In the comparison field we can check the values. We made multiple datasets with values removed to see which values would affect the accuracy of the dataset. We will be using the default settings for KNN and RandomForest in the Experimenter.

Used datasets: 1. bird_huml: contains only class attribute and huml attribute. 2. bird_huml_ulna_feml: contains only class attribute type and attributes huml, ulna and feml. 3. bird_huml_rm_ tarl_feml: contains class attribute type and removed attributes tarl and feml. 4. bird_huml_rm_tarl_femw_tibw: contains class attribute type and removed attributes tarl, femw and tibw. 5. bird_huml_rm_tarl: contains class attribute type and removed tarl. 6. bird_clean: contains all attributes.

Used Settings: 1. lazy.IBk, KNN = 3; 2. trees.RandomForest, depth = 10; 3. Simple Logistic, default (Because using 10-fold already uses automatic attribute selection.

```r
# Load results into R
weka_default <- read.csv("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 with default values")
```

Table 12: Quality metrics for each algorithm with default values (continued below)

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

Table 13: 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

The accuracy decreases the more attributes are removed this goes for all the models, therefore no attributes shall be removed.

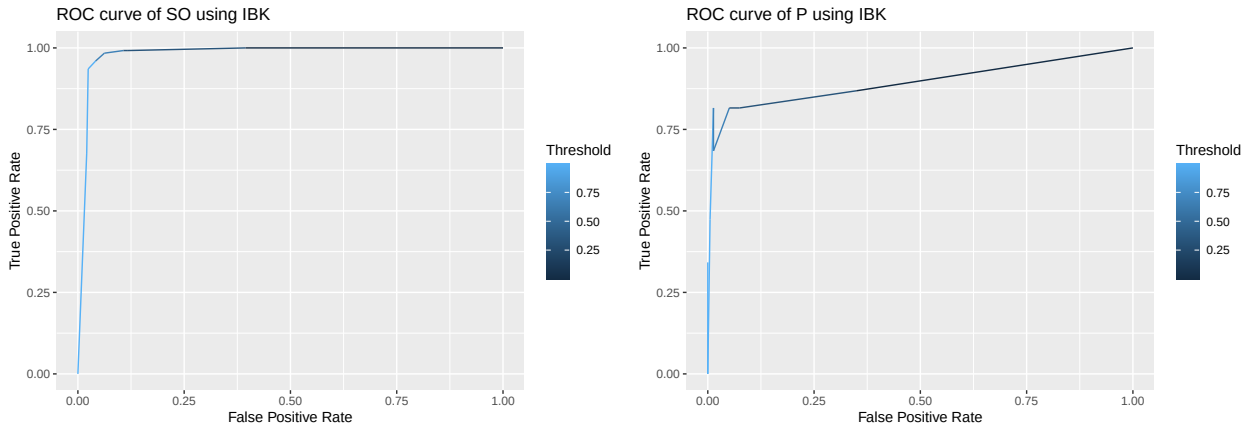
Final model

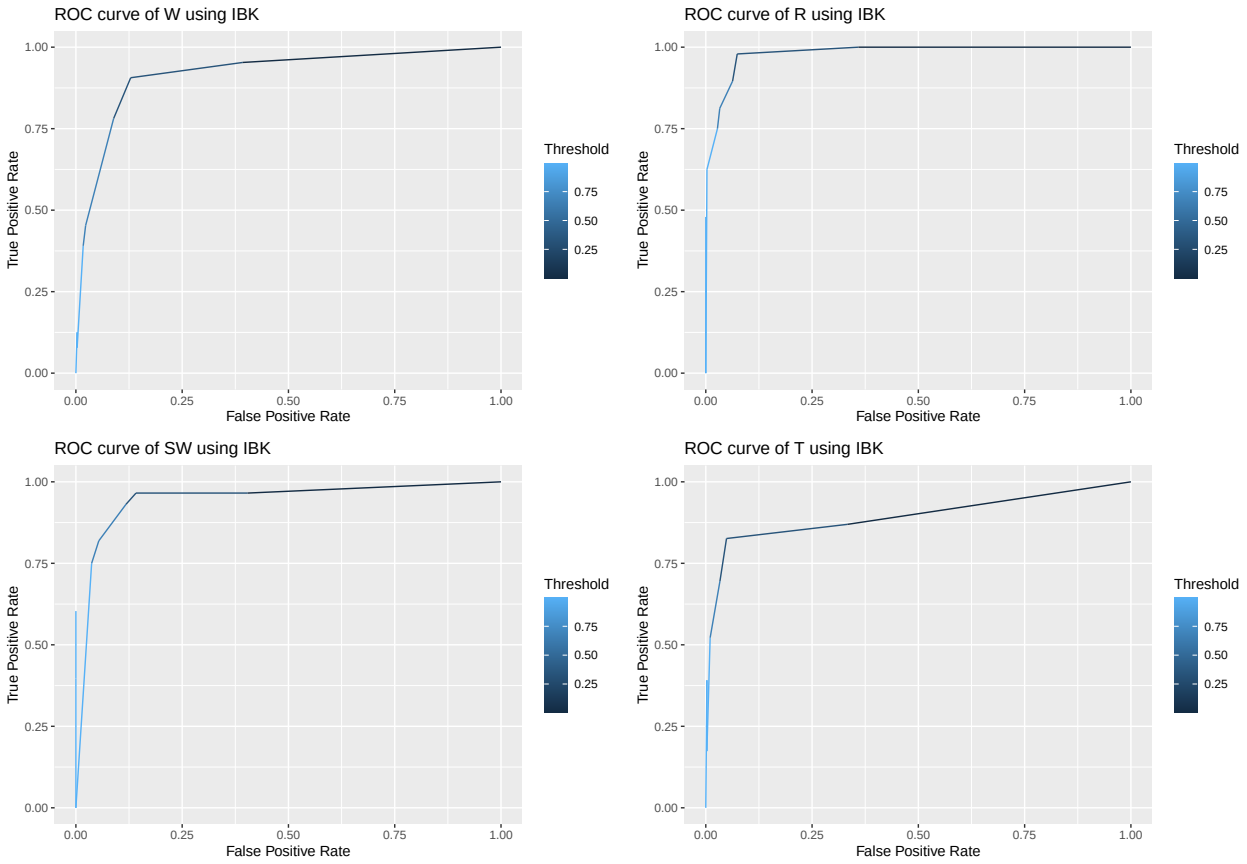
##ROC curve and confusion matrix For now it looks like the final model shall be a default Simple Logistic algorithm, with no parameters changed and no attributes removed from the dataset. We used the

clean_bird_data.csv file which contains no NA's but does contain outliers. To be sure of it, we shall inspect the ROC curves and confuson matrixes of the top 3 algorithms. The confusion matrixes were obtained via the Weka explorer using the classifier. Default options were used because 10-fold crossvalidation creates an automatic feature selection. We can also display our confusion matrix as an ROC curve to distill the most important quality metrics. To find the ROC curve we load our dataset into weka and go to the explorer, there we can go into the classify menu and use Logistic classifier to visualize the threshold curve. We do this for every attribute.

Pictured below are the ROC curves for IBK:

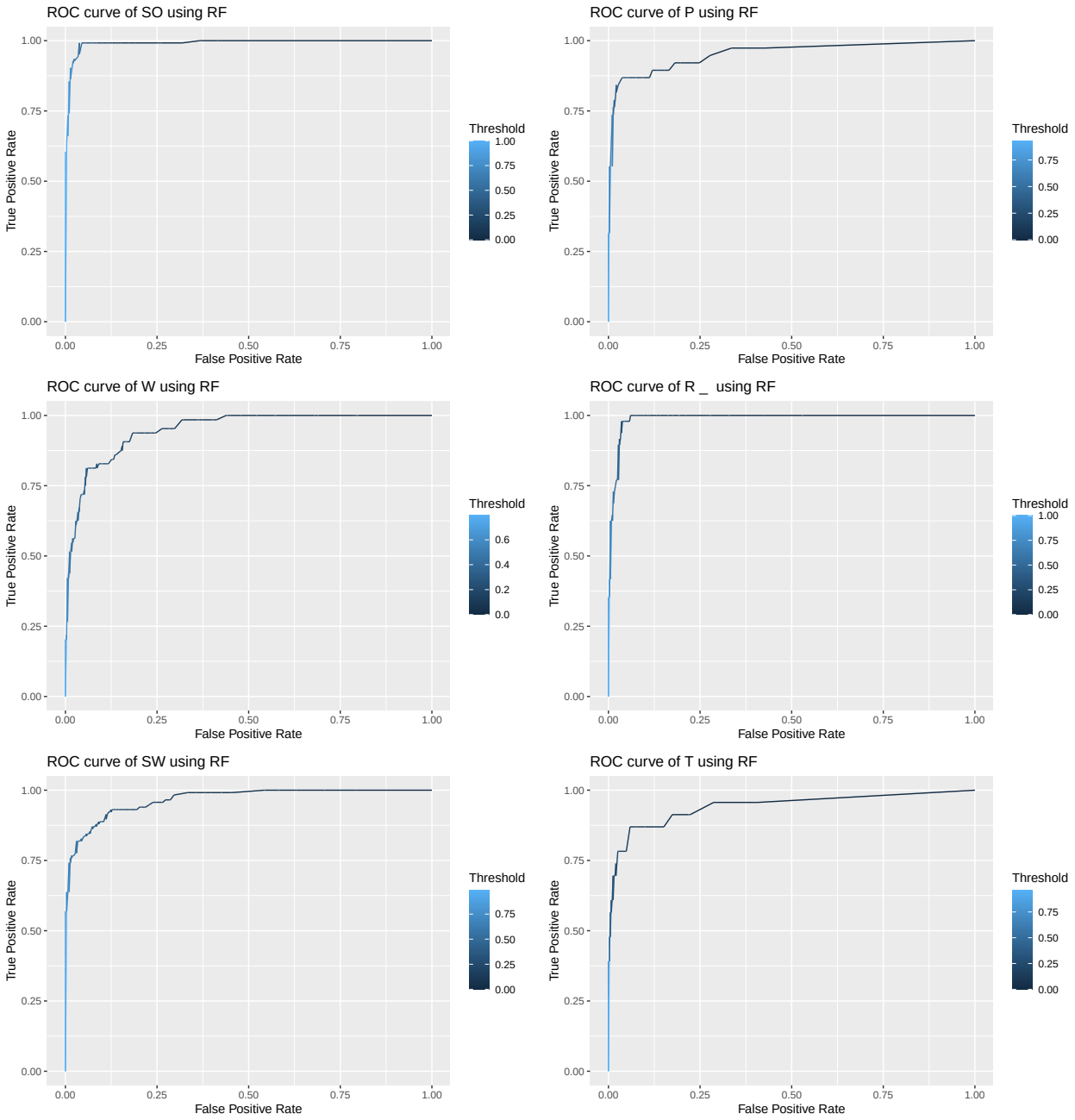
```
SO_curve_IBK <- read.arff("SO_curve_IBK.arff")
P_curve_IBK <- read.arff("P_curve_IBK.arff")
W_curve_IBK <- read.arff("W_curve_IBK.arff")
R_curve_IBK <- read.arff("R_curve_IBK.arff")
SW_curve_IBK <- read.arff("SW_curve_IBK.arff")
T_curve_IBK <- read.arff("T_curve_IBK.arff")
ggplot(SO_curve_IBK, aes(x=`False 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")
```





Pictured below are the values for RandomForest:

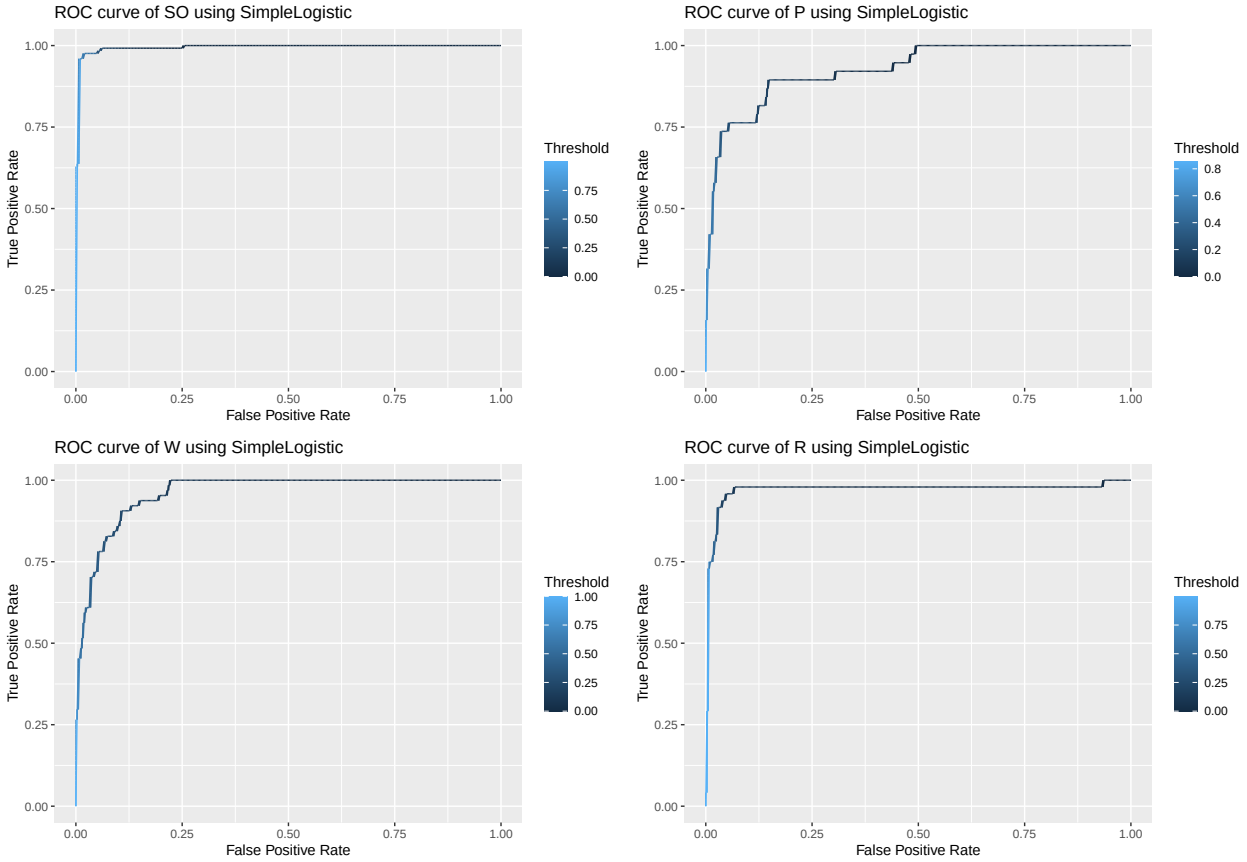
```
SO_curve_RF <- read.arff("SO_curve_RF.arff")
P_curve_RF <- read.arff("P_curve_RF.arff")
W_curve_RF <- read.arff("W_curve_RF.arff")
R_curve_RF <- read.arff("R_curve_RF.arff")
SW_curve_RF <- read.arff("SW_curve_RF.arff")
T_curve_RF <- read.arff("T_curve_RF.arff")
ggplot(SO_curve_RF, aes(x=`False Positive 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")
```

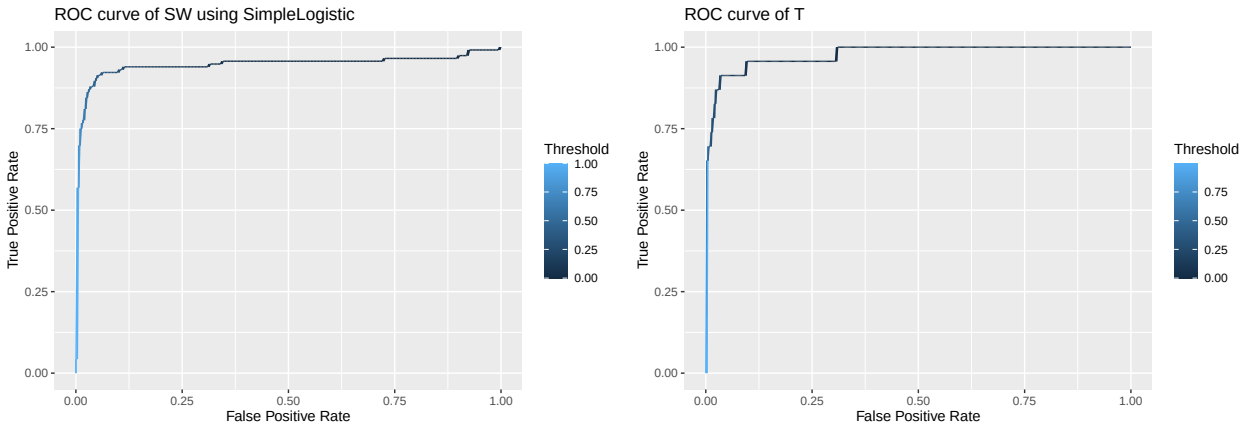


Pictured below are the values for the Simple Logistic:

```
SO_curve <- read.arff("SO_curve.arff")
P_curve <- read.arff("P_curve.arff")
W_curve <- read.arff("W_curve.arff")
R_curve <- read.arff("R_curve.arff")
SW_curve <- read.arff("SW_curve.arff")
T_curve <- read.arff("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")
```





Now for all the confusion matrixes:

Pictured below is the CM of SimpleLogsitic:

```
#Confusion matrix of SimpleLogistic
tabl_conf_matrix <- "
=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 103   6   1   1   1   4 |   a = SW
  13  40   2   5   0   4 |   b = W
   0   0  16   1   5   1 |   c = T
   1   2   0  41   3   1 |   d = R
   2   2   2   1  26   5 |   e = P
   0   1   0   0   1 122 |   f = SO
"
cat(tabl_conf_matrix)
```

```
##
## === Confusion Matrix ===
##
##    a   b   c   d   e   f   <-- classified as
##  103   6   1   1   1   4 |   a = SW
##   13  40   2   5   0   4 |   b = W
##    0   0  16   1   5   1 |   c = T
##    1   2   0  41   3   1 |   d = R
##    2   2   2   1  26   5 |   e = P
##    0   1   0   0   1 122 |   f = SO
```

Pictured below is the confusion matrix of RF:

```
#confusion matrix for RF
tabl_conf_matrix <- "
=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 100   5   2   4   0   5 |   a = SW
  17  35   1   3   5   3 |   b = W
   1   0  15   2   3   2 |   c = T
   4   0   0  44   0   0 |   d = R
   0   0   1   2  32   3 |   e = P
   0   0   0   1   0 123 |   f = SO
```

```
"
cat(tabl_conf_matrix)
```

```
##
## === Confusion Matrix ===
##
##    a   b   c   d   e   f   <-- classified as
##  100   5   2   4   0   5 |   a = SW
##   17  35   1   3   5   3 |   b = W
##    1   0  15   2   3   2 |   c = T
##    4   0   0  44   0   0 |   d = R
##    0   0   1   2  32   3 |   e = P
##    0   0   0   1   0 123 |   f = SO
```

Pictured below is the confusion matrix of IBK:

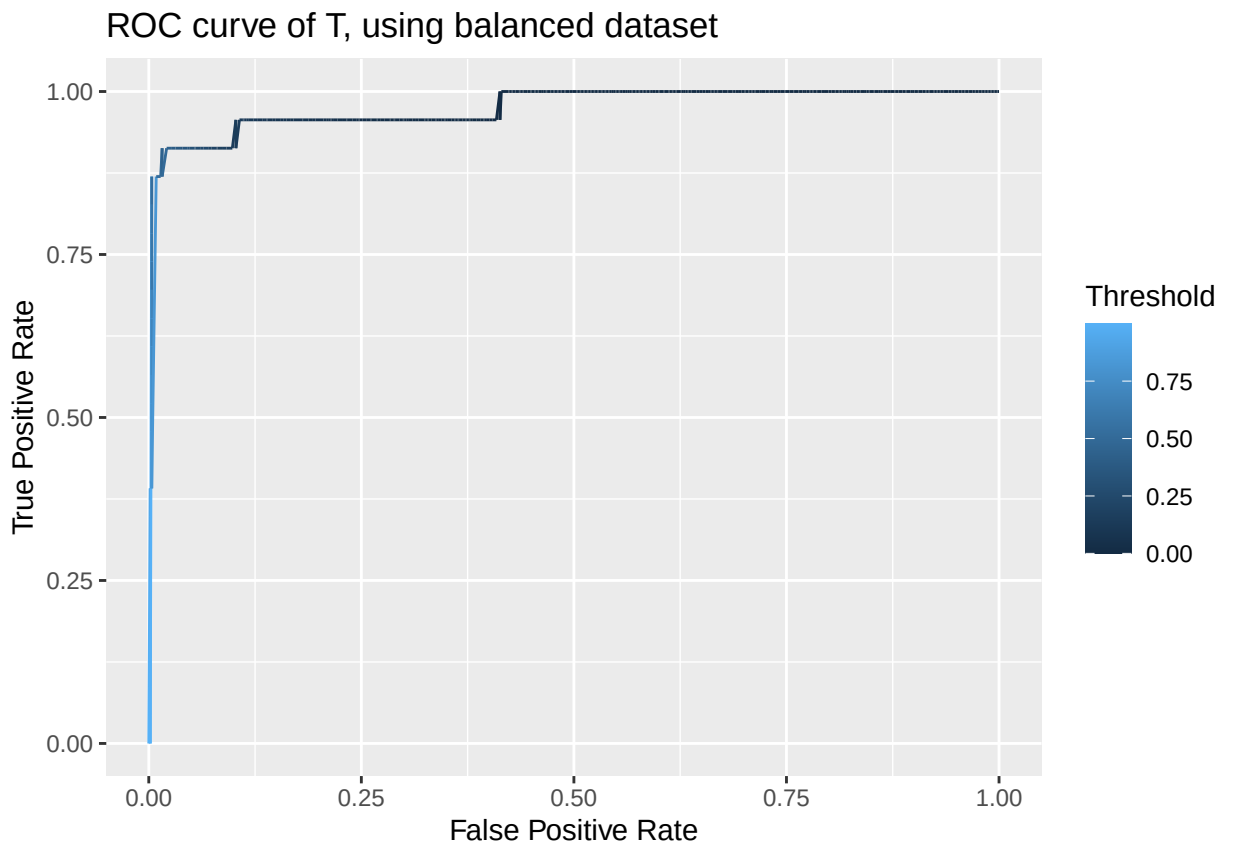
```
#confusion matrix for KNN
tabl_conf_matrix <- "
=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
  98   8   2   4   0   4 |   a = SW
  19  30   1   4   4   6 |   b = W
   4   0  13   3   1   2 |   c = T
   2   3   0  41   0   2 |   d = R
   0   0   2   1  31   4 |   e = P
   0   0   2   0   0 122 |   f = SO
"
cat(tabl_conf_matrix)
```

```
##
## === Confusion Matrix ===
##
##    a   b   c   d   e   f   <-- classified as
##   98   8   2   4   0   4 |   a = SW
##   19  30   1   4   4   6 |   b = W
##    4   0  13   3   1   2 |   c = T
##    2   3   0  41   0   2 |   d = R
##    0   0   2   1  31   4 |   e = P
##    0   0   2   0   0 122 |   f = SO
```

It appears our ROC curves are quite good for the larger groups of birds like SW and SO, this is because they are over represented in the dataset. The thresholds are also incomplete for certain groups. This can be explained by the low amount of data for T, P and W. Overall, the RandomForest algorithm and the SimpleLogistic algorithm have good looking ROC curves and confusion matrixes with low false positives and false negatives. Because SimpleLogistic has slightly higher accuracy and F1 score this will be the final algorithm.

It should be noted however, that the threshold is lower for certain labels. Let's apply a class balancing filter over our clean_bird_dataset to see if anything changed. We will use our type with the lowest instances (T) from our SimpleLogistic algorithm:

```
T_curve_classbalaner <- read.arff("T_curve_classbalancer.arff")
ggplot(T_curve_classbalaner, aes(x=`False Positive Rate`,
                                 y=`True Positive Rate`,
                                 colour = Threshold)) + geom_line() +
  labs(title= "ROC curve of T, using balanced dataset")
```



There is a slight change in the angle of the curve. It's false positive rate has decreased and true positive rate has increased. However, the line itself is still interrupted. The threshold doesn't fully go from 0 to 1, instead stopping at 1.0, even though the colors are present in the line itself. We can conclude that there simply isn't enough data to fully and accurately classify the data. However, based on previous research we will still choose SimpleLogistic as best fitting algorithm for this specific classification problem. This is mainly because as it has the overall quality metrics like F1 score, which is most informative for an unbalanced dataset.

Conclusion

Our final model will be a default SimpleLogistic algorithm, using an unbalanced dataset with no attributes removed. This has proven to be the most reliable algorithm next to RandomForest. However due to better quality metrics like F1 score, the algorithm that comes out on top is SimplpeLogistic.

We can also answer our research question; Using machine learning algorithms, which bone measurements are most informative for classifying birds into ecological groups? We can conclude that there is a significant relationship between bird bone measurements and their ecological groups, and bone measurements are informative for accurate classification. The most informative bones are the humerus and the ulna, these are located in the wing. However, all bone measurements contribute to a better overall classification.