CodeImplementation3

Aishwarya Saibewar

5/5/2023

```
#Load the required libraries
library(keras)
```

```
## Warning: package 'keras' was built under R version 4.1.2
```

```
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.1.2
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(abind)
library(tools)
```

Predicting bird species based on Sound

#Binary Classfication

```
# Set the working directory
setwd("/Users/aishwaryasaibewar/Documents/SeattleUniversity-MSDS/Courses/SU Course Work/SPRING_2023/Sta


#The folder includes 2 .dat files of amecro and barswa bird species
folder_path <- "/Users/aishwaryasaibewar/Documents/SeattleUniversity-MSDS/Courses/SU Course Work/SPRING_

#.dat files list
```

```
dat_files <- list.files(folder_path, pattern = ".dat")

# List of spectrogram data of all bird species
data_list <- list()

# List of names of all bird species
speciesname_list <- list()

#List of 1st dimension for assigning labels
species_label<-list()

# Iterating through the files and reading and proocessing them
for (i in 1:length(dat_files)) {
  data <- load(dat_files[i]) #Load the data from the dat files
  data <- species  #Initial data is a large array and the data is stored in species object
  data_list[[i]] <- data #Store the spectrogram data of each species in a list
  fname_without_exten <- file_path_sans_ext(dat_files[i]) #extract the names of each species
  speciesname_list[[i]] <- fname_without_exten #Then append it to the list and store it
  species_label[i]<-dim(species)[1] #Choose the 1st dimension for assigning labels to each species

}
```

```
# Using abind function to combine data stored in list for all species along a particular axis.It will b
data_array <- abind(data_list, along = 1)

#Check Dimension of the data
dim(data_array)
```

```
## [1] 107 343 256
```

```
#Assigning labels for the species, 0 to amecro and 1 to barswa
labels_assigned <- c(rep(0, species_label[1]), rep(1, species_label[2]))
```

```
#One-hot encode the labels
one_hot_labels <- array(labels_assigned,dim=c(length(labels_assigned),1))
```

```
# The dataset was divided into training and testing, with 70% of the data used as a training dataset to
set.seed(1)
train <- sample(nrow(data_array), 0.7 * nrow(data_array))

x_train <- data_array[train,,]
y_train <- one_hot_labels[train]

x_test <- data_array[-train,,]
y_test <- one_hot_labels[-train]

y_train <- matrix(y_train, ncol = 1)
y_test <- matrix(y_test, ncol = 1)
```

```
#Dimensions of test and training
dim(x_train)
```

```
## [1]  74 343 256
```

```
dim(x_test)
```

```
## [1]  33 343 256
```

```
dim(y_train)
```

```
## [1] 74  1
```

```
dim(y_test)
```

```
## [1] 33  1
```

```
#Reshape the data to a 4D array to feed as an input to the neural network
x_train <- array_reshape(x_train, c(dim(x_train)[1], dim(x_train)[2], dim(x_train)[3], 1))
x_test <- array_reshape(x_test, c(dim(x_test)[1], dim(x_test)[2], dim(x_test)[3], 1))
```

```
#Dimensions of test and training
dim(x_train)
```

```
## [1]  74 343 256   1
```

```
dim(x_test)
```

```
## [1]  33 343 256   1
```

```
# Build a Neural Network
binary_model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3),padding = "same", activation = "relu", input_shape =
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3, 3),padding = "same", activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  #layer_dropout(rate = 0.2) %>% #Adding dropout layer
  layer_flatten() %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dropout(rate = 0.3) %>% #Adding dropout layer
  layer_dense(units = 1, activation = "sigmoid")
```

```
## Loaded Tensorflow version 2.4.1
```

```
# Compile model by considering loss as binary_crossentropy and optimizer as adam and performance metric
binary_model %>% compile(loss = "binary_crossentropy",
  optimizer = "adam",
  metrics = list("accuracy")
)
```

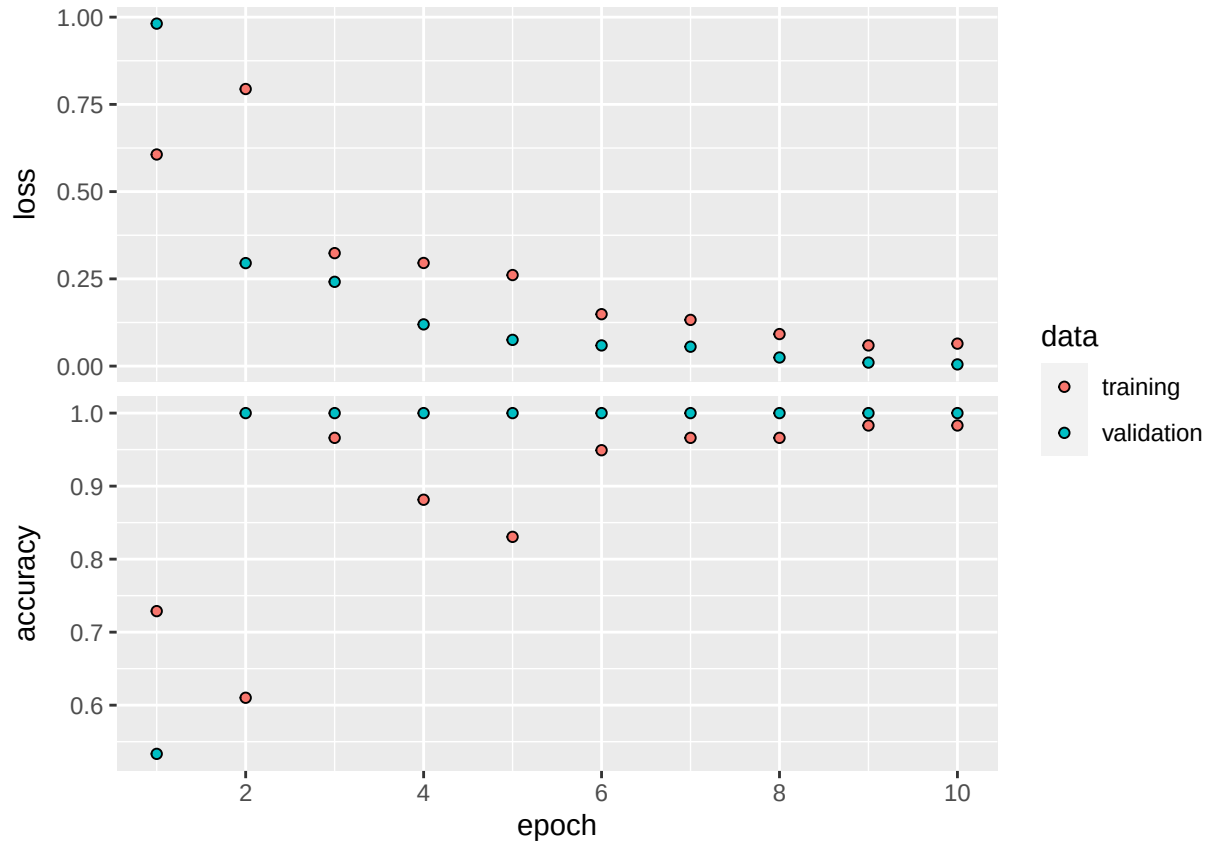
```r
#Summary of the binary classification model
summary(binary_model)
```

```
## Model: "sequential"
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## conv2d_1 (Conv2D)            (None, 343, 256, 32)      320
## _________________________________________________________________________
## max_pooling2d_1 (MaxPooling2D)  (None, 171, 128, 32)   0
## _________________________________________________________________________
## conv2d (Conv2D)              (None, 171, 128, 64)      18496
## _________________________________________________________________________
## max_pooling2d (MaxPooling2D)  (None, 85, 64, 64)       0
## _________________________________________________________________________
## flatten (Flatten)            (None, 348160)            0
## _________________________________________________________________________
## dense_1 (Dense)              (None, 32)                11141152
## _________________________________________________________________________
## dropout (Dropout)            (None, 32)                0
## _________________________________________________________________________
## dense (Dense)                (None, 1)                 33
## =========================================================================
## Total params: 11,160,001
## Trainable params: 11,160,001
## Non-trainable params: 0
## _________________________________________________________________________
```

```r
# Fit the model and estimate the time taken to train the model.
system.time(
   history <- binary_model %>%
      fit(x_train, y_train, epochs = 10, batch_size = 25,
        validation_split = 0.2)
 )
```

```
##    user  system elapsed
## 162.689  40.566  38.643
```

```r
#Plot the history of the model.
plot(history, smooth = FALSE, main = "Performance metrics for Binary Classification")
```




```
#Verify the dimensions
dim(x_test)
```

```
## [1]  33 343 256   1
```

```
dim(y_test)
```

```
## [1] 33  1
```

```
# Evaluate model on test data
test_binary_accuracy <- binary_model %>% evaluate(x_test, y_test)
cat("Test accuracy for binary classification model using CNN is ", test_binary_accuracy)
```

```
## Test accuracy for binary classification model using CNN is  0.04085369 0.969697
```

A binary classification model was developed to classify the bird species as amecro or barswa based on their sound. A Convolution Neural Network was developed to perform this classification. As shown in Figure 1 the model's loss and accuracy are plotted against the increasing number of epochs for the training and validation dataset. It can be observed that the accuracy of the training and validation data increases with the number of epochs, and the model converges at the 10th epoch with a batch size of 25. Similarly, the loss tends to decrease as the number of epochs increases. The initial model without dropout layers has predicted with a test accuracy of 93.3%. To improve the accuracy of the model, a dropout layer that ignored 30% of the training data was added to the CNN model. As a result of this, the accuracy of the test data has

slightly increased from 93.3% to 93.9%. Finally, the best model predicted the training data with an accuracy of 98%, validation data with an accuracy of 97%, and the test data with an accuracy of 96.9%. This model was able to classify the species more accurately because the sounds of the bird species amecro or barswa are quite different and the model was able to understand the sound patterns of the species and classify the new observations accordingly.

==========

MULTICLASS CLASSIFICATION

```
# Set the working directory
setwd("/Users/aishwaryasaibewar/Documents/SeattleUniversity-MSDS/Courses/SU Course Work/SPRING_2023/Sta


#The folder includes 2 .dat files of amecro and barswa bird species
folder_path <- "/Users/aishwaryasaibewar/Documents/SeattleUniversity-MSDS/Courses/SU Course Work/SPRING_

#.dat files list
dat_files <- list.files(folder_path, pattern = ".dat")

# List of spectrogram data of all bird species
data_list <- list()

# List of names of all bird species
speciesname_list <- list()

#List of 1st dimension for assigning labels
species_label<-list()

# Iterating through the files and reading and proocessing them
for (i in 1:length(dat_files)) {
  data <- load(dat_files[i]) #Load the data from the dat files
  data <- species  #Initial data is a large array and the data is stored in species object
  data_list[[i]] <- data #Store the spectrogram data of each species in a list
  fname_without_exten <- file_path_sans_ext(dat_files[i]) #extract the names of each species
  speciesname_list[[i]] <- fname_without_exten #Then append it to the list and store it
  species_label[i]<-dim(species)[1] #Choose the 1st dimension for assigning labels to each species

}
```

```
# Using abind function to combine data stored in list for all species along a particular axis.It will b
data_array <- abind(data_list, along = 1)

#Check Dimension of the data
dim(data_array)
```

```
## [1] 580 343 256
```

```
#Assigning labels for the 12 species
labels_assigned <- c(rep(0, species_label[1]), rep(1, species_label[2]),rep(2, species_label[3]),rep(3,
```

```r
# The dataset was divided into training and testing, with 70% of the data used as a training dataset to
set.seed(1)
train <- sample(nrow(data_array), 0.7 * nrow(data_array))

x_train <- data_array[train,,]
y_train <- to_categorical(labels_assigned[train])

x_test <- data_array[-train,,]
y_test <- to_categorical(labels_assigned[-train])
```

```r
#Check dimensions of test and training data
dim(x_train)
```

```
## [1] 406 343 256
```

```r
dim(x_test)
```

```
## [1] 174 343 256
```

```r
dim(y_train)
```

```
## [1] 406  12
```

```r
dim(y_test)
```

```
## [1] 174  12
```

```r
#Reshape the data to a 4D array to feed as an input to the neural network
x_train <- array_reshape(x_train, c(dim(x_train)[1], dim(x_train)[2], dim(x_train)[3], 1))
x_test <- array_reshape(x_test, c(dim(x_test)[1], dim(x_test)[2], dim(x_test)[3], 1))
```

```r
#Dimensions of test and training
dim(x_train)
```

```
## [1] 406 343 256   1
```

```r
dim(x_test)
```

```
## [1] 174 343 256   1
```

```r
# Build a Neural Network with 2 convolution layers,2 max pool layers, one flatten, 3 dense layers and a
multiclass_model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3), activation = "relu", input_shape = c(343,256,1)) %
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dropout(rate = 0.4) %>% #Added for overfitting
  layer_dense(units = 12, activation = "softmax")
```

```
# Compile model by considering loss as binary_crossentropy and optimizer as adam and performance metric
multiclass_model %>% compile(loss = "categorical_crossentropy",
  optimizer = "adam",
  metrics = list("accuracy")
)
```

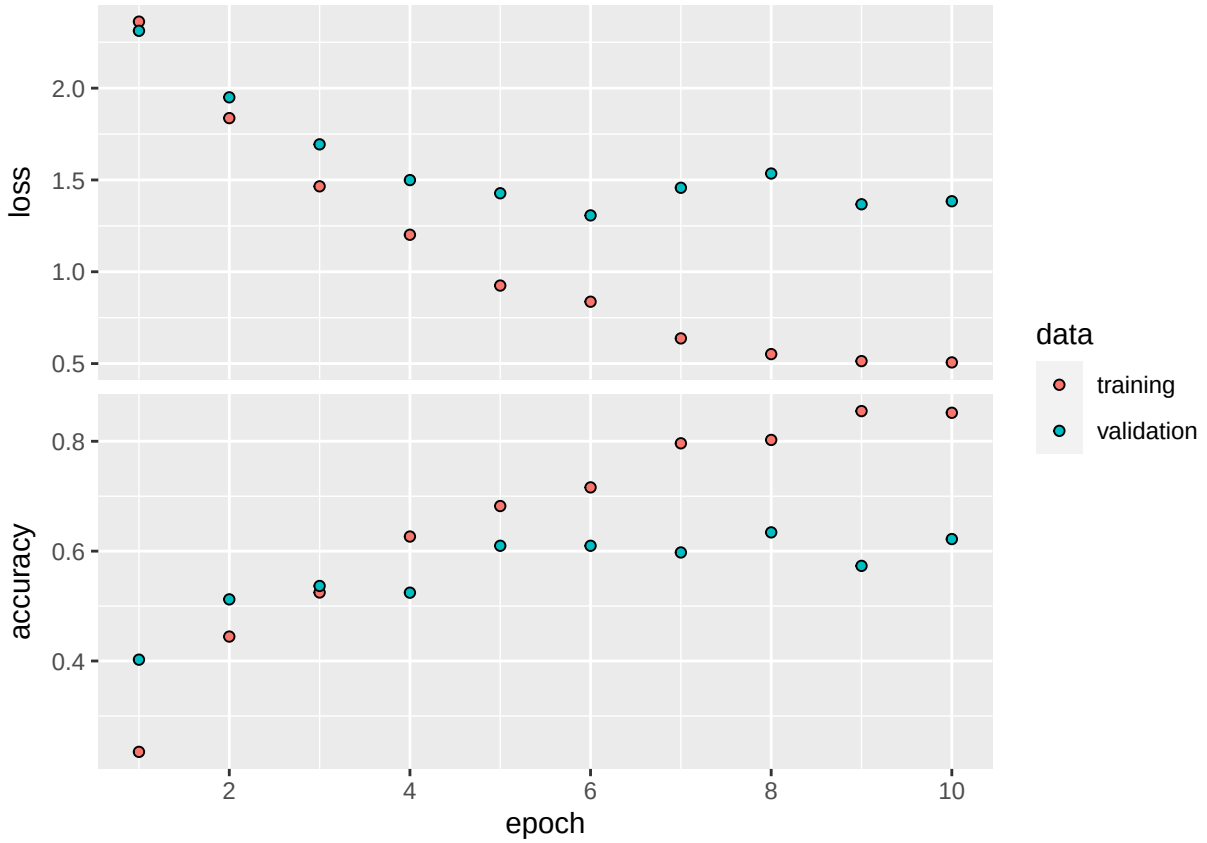
```
#Summary of the multiclass classification model
summary(multiclass_model)
```

```
## Model: "sequential_1"
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## conv2d_3 (Conv2D)            (None, 341, 254, 32)      320
## _________________________________________________________________________
## max_pooling2d_3 (MaxPooling2D)  (None, 170, 127, 32)   0
## _________________________________________________________________________
## conv2d_2 (Conv2D)            (None, 168, 125, 64)      18496
## _________________________________________________________________________
## max_pooling2d_2 (MaxPooling2D)  (None, 84, 62, 64)     0
## _________________________________________________________________________
## flatten_1 (Flatten)          (None, 333312)            0
## _________________________________________________________________________
## dense_4 (Dense)              (None, 64)                21332032
## _________________________________________________________________________
## dense_3 (Dense)              (None, 32)                2080
## _________________________________________________________________________
## dropout_1 (Dropout)          (None, 32)                0
## _________________________________________________________________________
## dense_2 (Dense)              (None, 12)                396
## =========================================================================
## Total params: 21,353,324
## Trainable params: 21,353,324
## Non-trainable params: 0
## _________________________________________________________________________
```

```
# Fit the model and estimate the time taken to train the model.
system.time(
   history <- multiclass_model %>%
      fit(x_train, y_train, epochs = 10, batch_size = 25,
        validation_split = 0.2)
 )
```

```
##     user   system  elapsed
## 1016.035  256.451  237.463
```

```
#Plot the history
plot(history, smooth = FALSE)
```




```
#Check the dimensions of test and training
dim(x_test)
```

```
## [1] 174 343 256   1
```

```
dim(y_test)
```

```
## [1] 174  12
```

```
# Evaluate model on test data
test_multi_accuracy <- multiclass_model %>% evaluate(x_test, y_test)
cat("Test accuracy for multiclass classification model using CNN is ", test_multi_accuracy)
```

```
## Test accuracy for multiclass classification model using CNN is  1.297022 0.683908
```

A multiclass classification model was developed to classify the 12 bird species based on their sound using a Convolution Neural Network. As shown in Figure 2 the model's loss and accuracy are plotted against the increasing number of epochs. As shown in Figure 2, the model's accuracy and loss are plotted against the training and validation datasets for the increasing number of epochs. It can be observed that the model converges at the 10th epoch with a batch size of 25, and the accuracy of the training and validation data increases with the number of epochs. And the loss tends to decrease as the number of epochs increases. The initial model without dropout layers has predicted the training data, validation data, and test data with an accuracy of 90.8%, 70.1%, and 56.3% respectively. It can be seen that the model is overfitting the training

data and is unable to generalize the patterns in test and validation datasets. To overcome this issue, an additional dropout layer that ignores 40% of the training data was added to the CNN model. This has resulted in decreasing the accuracy of training data from 90.8% to 82.8% and increasing the accuracy of test data from 56.3% to 67.2%.

```
#The names of the species
speciesname_list
```

```
## [[1]]
## [1] "amecro"
##
## [[2]]
## [1] "barswa"
##
## [[3]]
## [1] "bkcchi"
##
## [[4]]
## [1] "blujay"
##
## [[5]]
## [1] "daejun"
##
## [[6]]
## [1] "houfin"
##
## [[7]]
## [1] "mallar3"
##
## [[8]]
## [1] "norfli"
##
## [[9]]
## [1] "rewbla"
##
## [[10]]
## [1] "stejay"
##
## [[11]]
## [1] "wesmea"
##
## [[12]]
## [1] "whcspa"
```

```
# Predict the model
predict <- multiclass_model %>% predict(x_test)
colnames(predict) <- speciesname_list
head(predict)
```

```
##         amecro       barswa       bkcchi       blujay       daejun       houfin
## [1,] 0.9997925 4.517176e-06 1.915617e-06 5.047864e-08 1.076086e-04 2.138709e-05
## [2,] 0.9999909 4.274308e-07 2.081657e-08 2.068129e-09 2.571410e-06 1.890889e-06
## [3,] 0.9997440 1.297887e-05 3.757193e-05 2.051244e-07 7.234476e-05 1.326729e-05
```

```
## [4,] 0.9937984 2.060163e-04 4.995524e-04 1.824290e-05 8.574291e-04 4.871043e-04
## [5,] 0.9995331 2.773515e-05 5.978975e-06 4.587525e-07 1.834058e-04 8.928256e-05
## [6,] 0.9106319 4.406515e-03 1.816162e-03 3.082101e-04 5.581217e-04 5.430363e-03
##           mallar3       norfli       rewbla       stejay       wesmea
## [1,] 3.261410e-06 2.205689e-07 3.824868e-05 2.434721e-05 3.947842e-06
## [2,] 2.399539e-06 4.018204e-08 1.064326e-06 5.290655e-07 2.209569e-08
## [3,] 2.196760e-05 2.941223e-06 7.865383e-05 1.009489e-05 3.685392e-06
## [4,] 7.109014e-04 1.111504e-04 1.126674e-03 1.506087e-03 3.947592e-04
## [5,] 2.992274e-05 1.352223e-06 2.537644e-05 9.550908e-05 2.807312e-06
## [6,] 3.549807e-02 5.711262e-04 2.143132e-03 2.960632e-02 1.247055e-03
##            whcspa
## [1,] 1.917262e-06
## [2,] 5.815135e-08
## [3,] 2.212154e-06
## [4,] 2.835986e-04
## [5,] 5.099837e-06
## [6,] 7.783043e-03
```

```
#Ensure that the sum of the probbabilities is equal to 1
sum(predict[1, ])
```

```
## [1] 0.9999999
```

```
sum(predict[2, ])
```

```
## [1] 1
```

```
# https://www.rdocumentation.org/packages/nnet/versions/7.3-12/topics/predict.nnet

#Fetch the actual and predicted values
Predicted = max.col(predict) - 1
Actual = max.col(y_test) - 1
```

```
#Confusion matrix for multiclass classification
multi_data_cf <- table(Predicted, Actual)
multi_data_cf
```

```
##           Actual
## Predicted  0  1  2  3  4  5  6  7  8  9 10 11
##        0  12  0  2  0  0  0  0  0  0  0  0  0
##        1   0 14  0  0  0  2  0  0  0  0  0  1
##        2   0  0  5  0  0  0  0  0  0  0  0  0
##        3   0  1  0  7  0  0  0  0  3  1  1  0
##        4   0  1  2  1 17  0  0  0  1  0  0  4
##        5   0  0  0  2  0  6  0  0  1  0  2  0
##        6   0  0  0  0  0  0  4  0  0  0  0  0
##        7   0  1  2  2  3  1  0 17  0  0  1  1
##        8   0  0  0  1  0  2  0  0 11  0  0  0
##        9   0  0  0  2  0  0  1  0  0  6  0  0
##        10  0  0  0  0  0  1  0  0  1  0  7  0
##        11  0  1  0  1  0  1  8  0  0  0  0 13
```

```
#Rename the confusion matrix
colnames(multi_data_cf) <- c("amecro","barswa","bkcchi","blujay","daejun","houfin","mallar3","norfli","
rownames(multi_data_cf) <- c("amecro","barswa","bkcchi","blujay","daejun","houfin","mallar3","norfli","
```

```
#Final Confusion matrix for multiclass classification
multi_data_cf
```

```
##           Actual
## Predicted amecro barswa bkcchi blujay daejun houfin mallar3 norfli rewbla
##   amecro      12      0      2      0      0      0       0      0      0
##   barswa       0     14      0      0      0      2       0      0      0
##   bkcchi       0      0      5      0      0      0       0      0      0
##   blujay       0      1      0      7      0      0       0      0      3
##   daejun       0      1      2      1     17      0       0      0      1
##   houfin       0      0      0      2      0      6       0      0      1
##   mallar3      0      0      0      0      0      0       4      0      0
##   norfli       0      1      2      2      3      1       0     17      0
##   rewbla       0      0      0      1      0      2       0      0     11
##   stejay       0      0      0      2      0      0       1      0      0
##   wesmea       0      0      0      0      0      1       0      0      1
##   whcspa       0      1      0      1      0      1       8      0      0
##           Actual
## Predicted stejay wesmea whcspa
##   amecro       0      0      0
##   barswa       0      0      1
##   bkcchi       0      0      0
##   blujay       1      1      0
##   daejun       0      0      4
##   houfin       0      2      0
##   mallar3      0      0      0
##   norfli       0      1      1
##   rewbla       0      0      0
##   stejay       6      0      0
##   wesmea       0      7      0
##   whcspa       0      0     13
```

The confusion matrix of the multiclass classification model is shown in Figure 3. The diagonal elements represent the correct classifications and the others represent the misclassifications. It very well may be seen from the confusion matrix that there are misclassifications of the observations. Some of the incorrect classifications include barswa being predicted as houfin and whcspa, bkcchi as barswa and daejun, and norfli as blujay and daejun. This could be due to the similarity of these bird species sounds. Additionally, the spectrograms used for the analysis included data on just the louder parts of the audio clip. Although important patterns of the species may be present in the clip's louder parts, some species that had their identity in the clip's softer parts may have been incorrectly classified.