ANN(ABIGAIL)

2024-07-31

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
# Installation of necessary packages and libraries

install.packages(c('neuralnet','keras','tensorflow'), dependencies = T)
```

```
## Installing packages into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(neuralnet)
install.packages("tidyverse")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::compute() masks neuralnet::compute()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
#loading and viewing dataset

iris<-iris %>%mutate_if(is.character, as.factor)

# Get the total number of rows in the dataset
total_rows <- nrow(iris)
total_rows
```

```
## [1] 150
```

```
# View the first 10 rows of the dataset
head(iris, 10)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
```

```
## 1           5.1         3.5          1.4         0.2  setosa
## 2           4.9         3.0          1.4         0.2  setosa
## 3           4.7         3.2          1.3         0.2  setosa
## 4           4.6         3.1          1.5         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 7           4.6         3.4          1.4         0.3  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 10          4.9         3.1          1.5         0.1  setosa
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
# Train and test split

set.seed(254)
data_rows<-floor(0.80 * nrow(iris))
data_rows
```

```
## [1] 120
```

```
train_indices<-sample(c(1:nrow(iris)), data_rows)
train_indices
```

```
##   [1]  55  37 146  70  45 124  20  76 144   3  88  10 136 126 102 125  64 111
##  [19] 122  32 147 123  95 101 149 143  94 150  11  83  54  57  61  48  29  69
##  [37] 130 115 145  17  50  96  35  93  49  12  14  60  18  97 109 134  62 113
##  [55]  75 119  41  27  25  89 100  91  19 137  46 103  85   6  44  86  71  36
##  [73] 104  42 139 118 106   9  43  84  66  39   7  72 117 108   4  38 138  65
##  [91]   5   2  87  82  40  77 128  67  92 131  74  56  59 120  23  13  33 107
## [109] 127  24 116  34  68  58  73  80   8  99 121 133
```

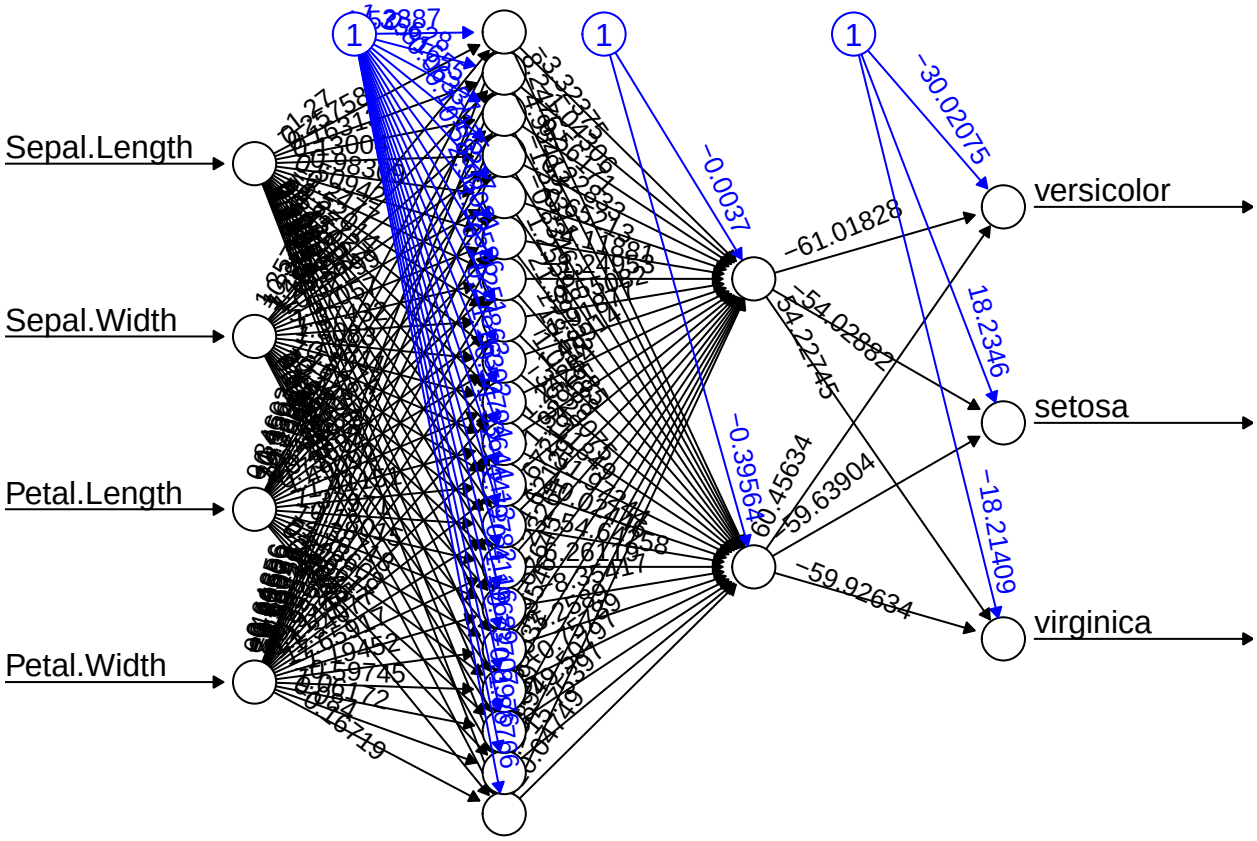
```
train_data<-iris[train_indices, ]
train_samples <- sample_n(train_data, 5)
train_samples
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          5.0         2.0          3.5         1.0 versicolor
## 2          5.7         3.8          1.7         0.3     setosa
## 3          5.3         3.7          1.5         0.2     setosa
## 4          7.1         3.0          5.9         2.1  virginica
## 5          5.2         3.4          1.4         0.2     setosa
```

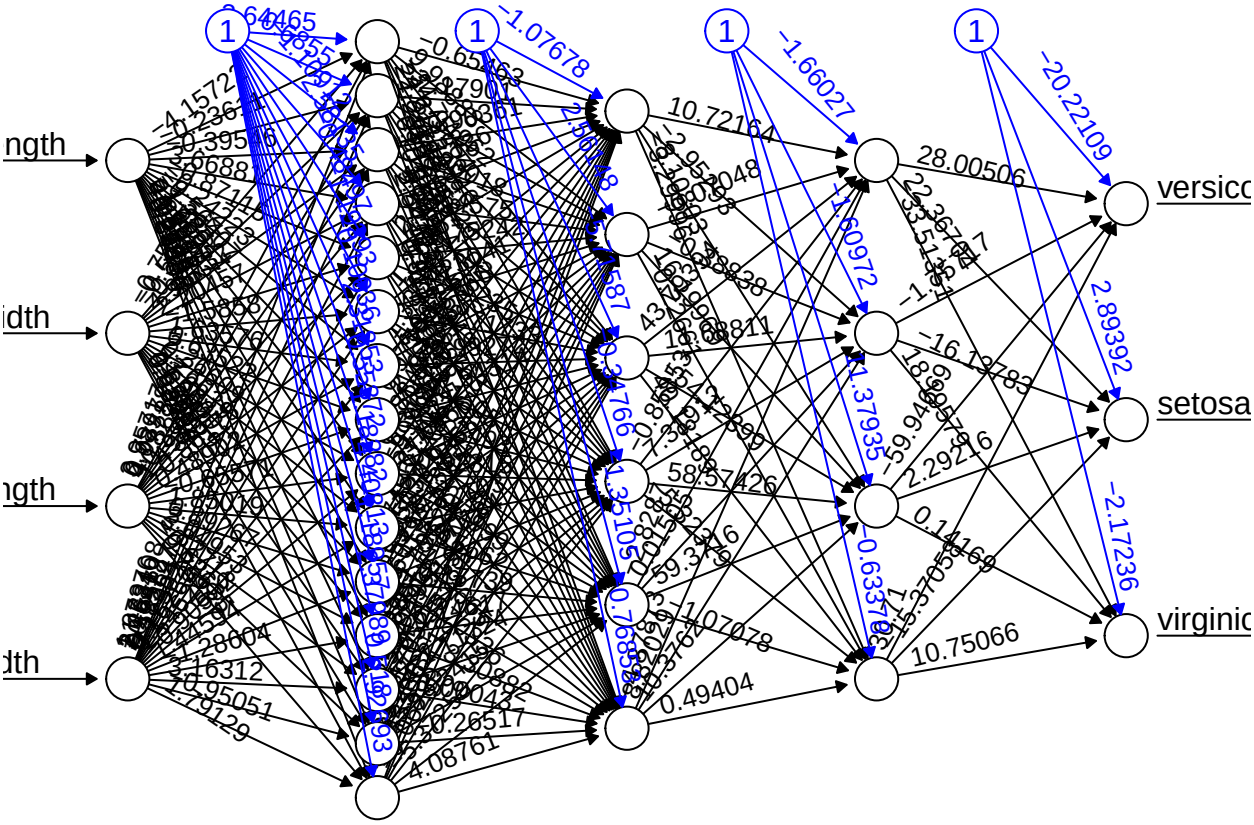
```
test_data<-iris[-train_indices,]
test_samples <- sample_n(test_data, 5)
test_samples
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          6.5         3.0          5.2         2.0  virginica
## 2          5.7         4.4          1.5         0.4     setosa
## 3          5.1         3.5          1.4         0.2     setosa
## 4          7.0         3.2          4.7         1.4 versicolor
## 5          6.7         3.1          5.6         2.4  virginica
```

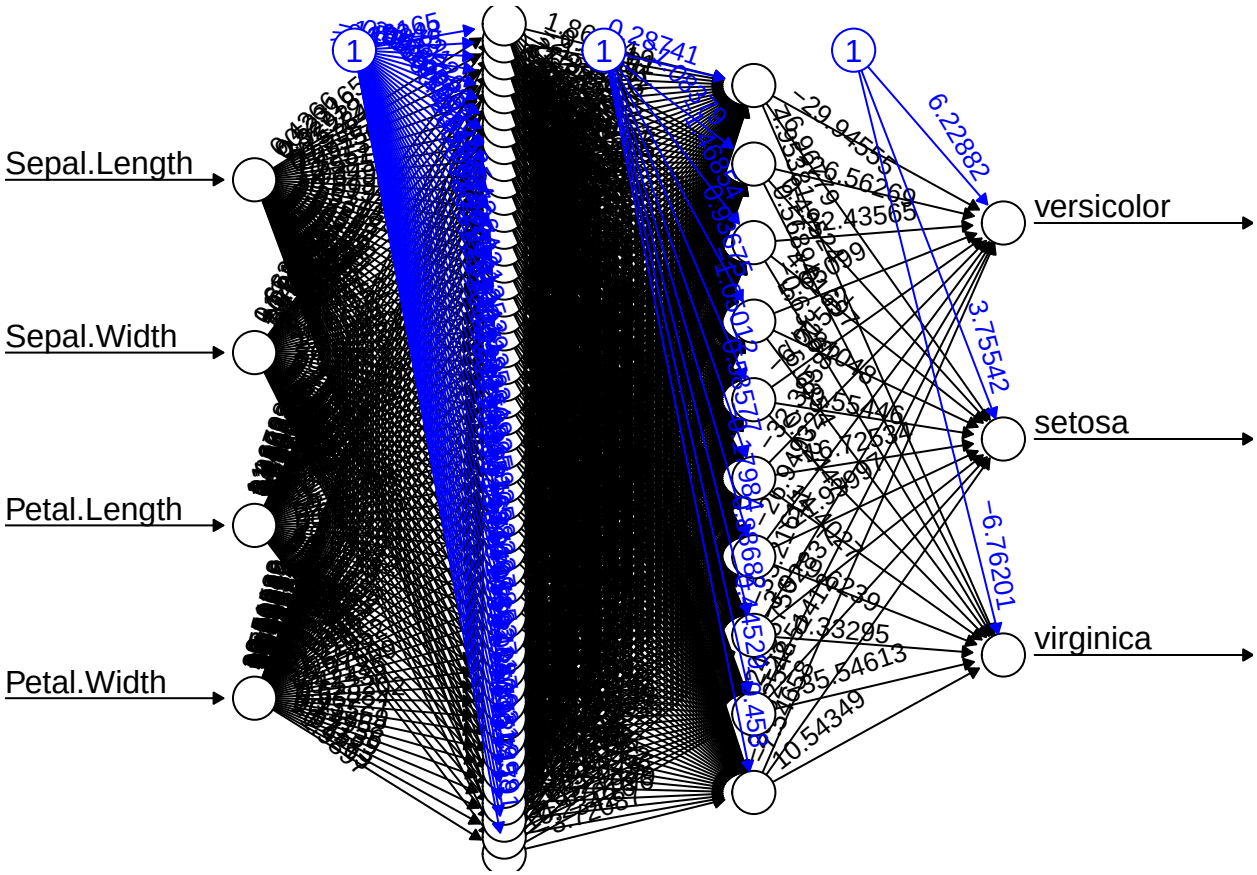
```
# Define and train the first model with hidden layers c(6, 2)
model1 <- neuralnet(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,data = train_data

# Define and train the second model with hidden layers c(10, 4)
model2 <- neuralnet(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,data = train_data

# Define and train the third model with hidden layers c(50,10)
model3 <- neuralnet(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,data = train_data

# Plot the models
plot(model1, rep = 'best', main = "Model with hidden layers c(20, 2)")
```




```
plot(model2, rep = 'best', main = "Model with hidden layers c(15,6, 4)")
```




```
plot(model3, rep = 'best', main = "Model with hidden layers c(50,10) ")
```




```
# Model evaluation
#predict categories - test dataset
#list of category names
#dataframe
# table - actual and predicated


# Function to predict and calculate accuracy
evaluate_model <- function(model, test_data) {
  pred <- predict(model, test_data)
  labels <- c("setosa", "versicolor", "virginica")

  prediction_label <- data.frame(max.col(pred)) %>%
    mutate(pred = labels[max.col.pred.]) %>%
    select(pred) %>%
    unlist()

  table(test_data$Species, prediction_label)

  check <- as.numeric(test_data$Species) == max.col(pred)
  accuracy <- (sum(check) / nrow(test_data)) * 100
  return(accuracy)
}
```

```r
# Evaluate each model
accuracy1 <- evaluate_model(model1, test_data)
accuracy2 <- evaluate_model(model2, test_data)
accuracy3 <- evaluate_model(model3, test_data)

# Print accuracies
print(paste("Accuracy of model 1 (20, 2):", accuracy1))
```

```
## [1] "Accuracy of model 1 (20, 2): 100"
```

```r
print(paste("Accuracy of model 2 (15,6, 4):", accuracy2))
```

```
## [1] "Accuracy of model 2 (15,6, 4): 93.3333333333333"
```

```r
print(paste("Accuracy of model 3 (50,10):", accuracy3))
```

```
## [1] "Accuracy of model 3 (50,10): 100"
```

```r
# Install the kableExtra package
install.packages("kableExtra")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
# Load necessary libraries
library(dplyr)
library(neuralnet)
library(knitr)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
##
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```r
# Create a data frame with the accuracies
results <- data.frame(
  Model = c("Model 1 (20, 2)", "Model 2 (15,6, 4)", "Model 3 (50,10)"),
  Accuracy = c(accuracy1, accuracy2, accuracy3)
)
# Print the table using knitr and kableExtra
knitr::kable(results, format = "latex", booktabs = TRUE, col.names = c("Model", "Accuracy (%)")) %>%
  kable_styling(latex_options = "hold_position", full_width = FALSE)
```

Model	Accuracy (%)
Model 1 (20, 2)	100.00000
Model 2 (15,6, 4)	93.33333
Model 3 (50,10)	100.00000

```r
# Install the pROC package if not already installed
install.packages("pROC")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
# Load necessary library
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
# Function to plot ROC curve and calculate AUC
plot_roc <- function(model, test_data) {
  pred <- predict(model, test_data)
  roc_curve <- roc(response = as.numeric(test_data$Species), predictor = as.numeric(max.col(pred)))
  auc_value <- auc(roc_curve)

  plot(roc_curve, main = paste("ROC Curve (AUC =", round(auc_value, 2), ")"))
}

# Plot ROC curves for each model
par(mfrow = c(1, 3))
plot_roc(model1, test_data)
```

```
## Warning in roc.default(response = as.numeric(test_data$Species), predictor =
## as.numeric(max.col(pred))): 'response' has more than two levels. Consider
## setting 'levels' explicitly or using 'multiclass.roc' instead
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```r
plot_roc(model2, test_data)
```

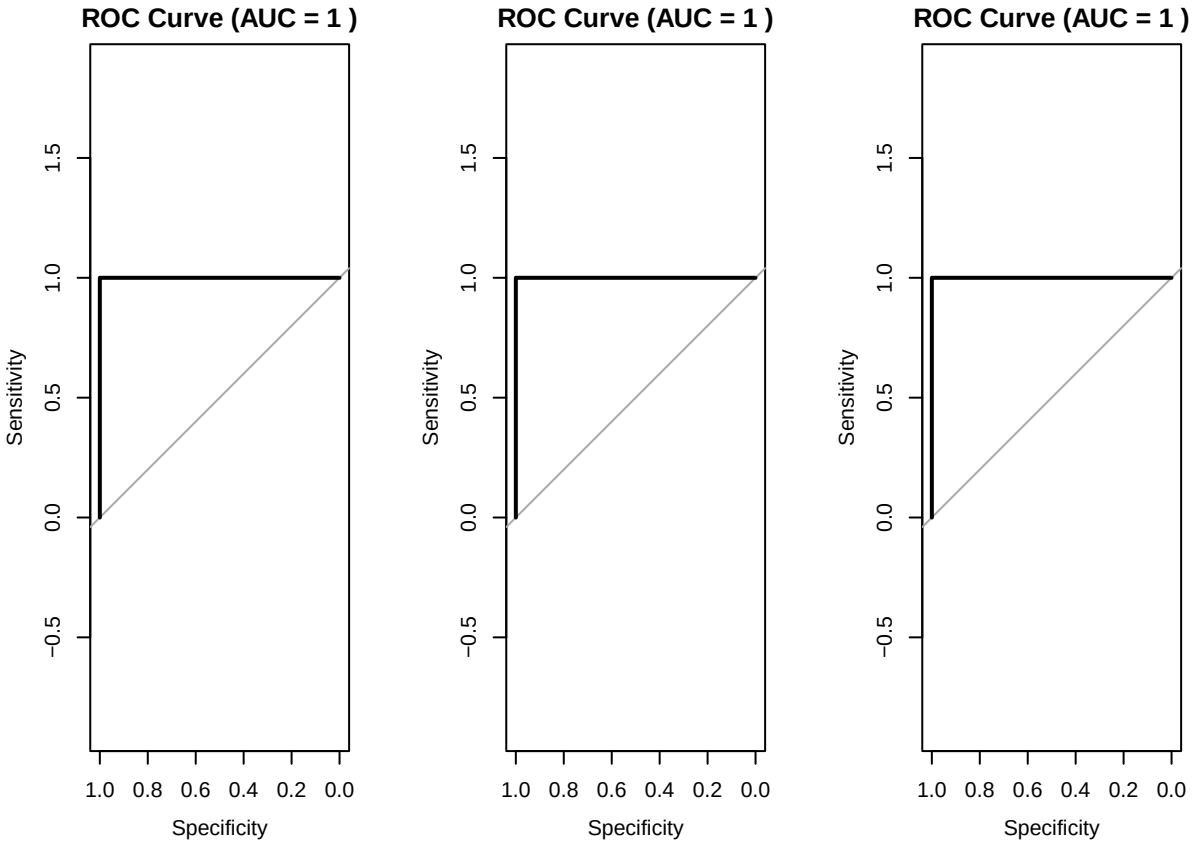
```
## Warning in roc.default(response = as.numeric(test_data$Species), predictor =
## as.numeric(max.col(pred))): 'response' has more than two levels. Consider
## setting 'levels' explicitly or using 'multiclass.roc' instead
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```r
plot_roc(model3, test_data)
```

```
## Warning in roc.default(response = as.numeric(test_data$Species), predictor =
## as.numeric(max.col(pred))): 'response' has more than two levels. Consider
## setting 'levels' explicitly or using 'multiclass.roc' instead
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```


ROC Curve (AUC = 1)
Sensitivity
1.5
1.0
0.5
0.0
−0.5
1.0 0.8 0.6 0.4 0.2 0.0
Specificity
ROC Curve (AUC = 1)
Sensitivity
1.5
1.0
0.5
0.0
−0.5
1.0 0.8 0.6 0.4 0.2 0.0
Specificity
ROC Curve (AUC = 1)
Sensitivity
1.5
1.0
0.5
0.0
−0.5
1.0 0.8 0.6 0.4 0.2 0.0
Specificity