report

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:

```
# Install necessary packages (run this in the console if not already installed)
install.packages(c('neuralnet', 'keras', 'tensorflow'), dependencies = TRUE)
```

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

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

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

```
library(neuralnet)
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
```

```
# Prepare the iris dataset
iris <- iris %>% mutate(across(where(is.character), as.factor))
print(sample_n(iris, 3))  # Print a random sample of the iris dataset
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          6.2         3.4          5.4         2.3  virginica
## 2          5.9         3.2          4.8         1.8 versicolor
## 3          4.8         3.0          1.4         0.3     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))
train_indices <- sample(seq_len(nrow(iris)), data_rows)
train_data <- iris[train_indices, ]
test_data <- iris[-train_indices, ]

# Ensure no overlap between train and test data
cat("Number of unique rows in train_data:\n")
```

```
## Number of unique rows in train_data:
```

```
print(nrow(unique(train_data)))
```

```
## [1] 119
```

```
cat("Number of unique rows in test_data:\n")
```

```
## Number of unique rows in test_data:
```

```
print(nrow(unique(test_data)))
```

```
## [1] 30
```

```
# Print random sample of train_data and test_data
cat("Random sample of train_data:\n")
```

```
## Random sample of train_data:
```

```
print(sample_n(train_data, 3))
```

```
##   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
```

```
cat("Random sample of test_data:\n")
```

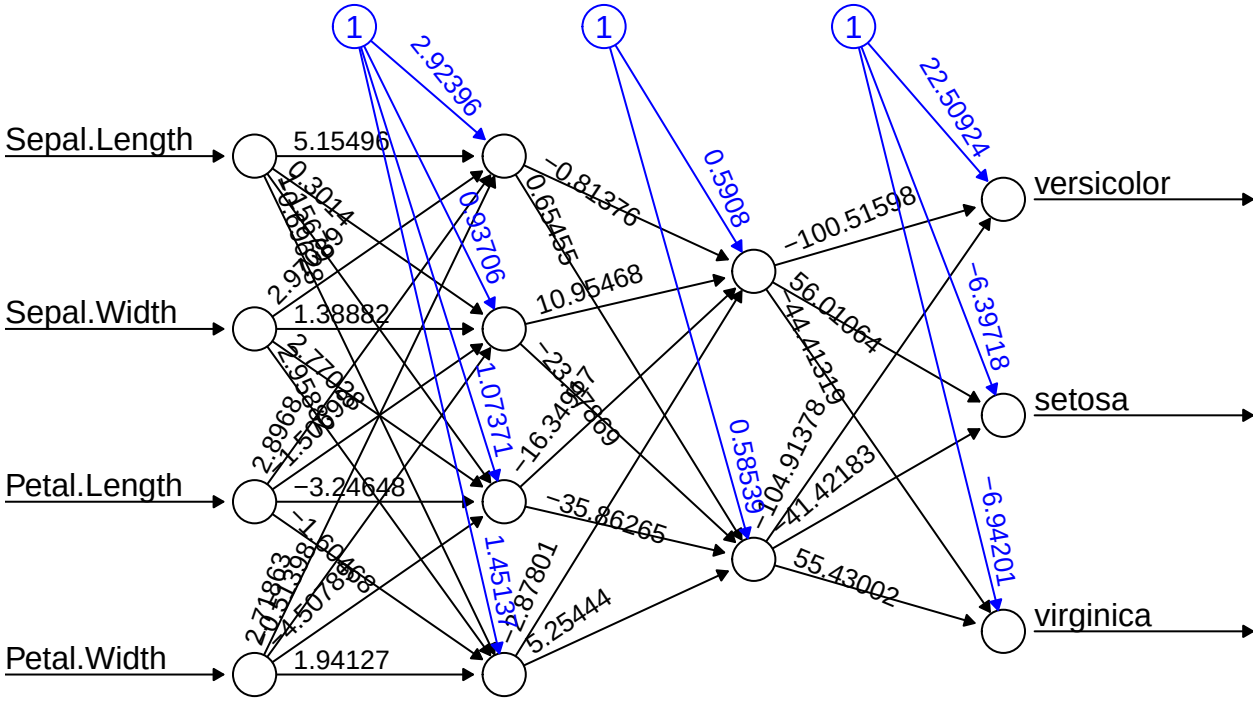
```
## Random sample of test_data:
```

```
print(sample_n(test_data, 3))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 1          5.8         4.0          1.2         0.2    setosa
## 2          5.7         4.4          1.5         0.4    setosa
## 3          6.9         3.1          5.1         2.3 virginica
```

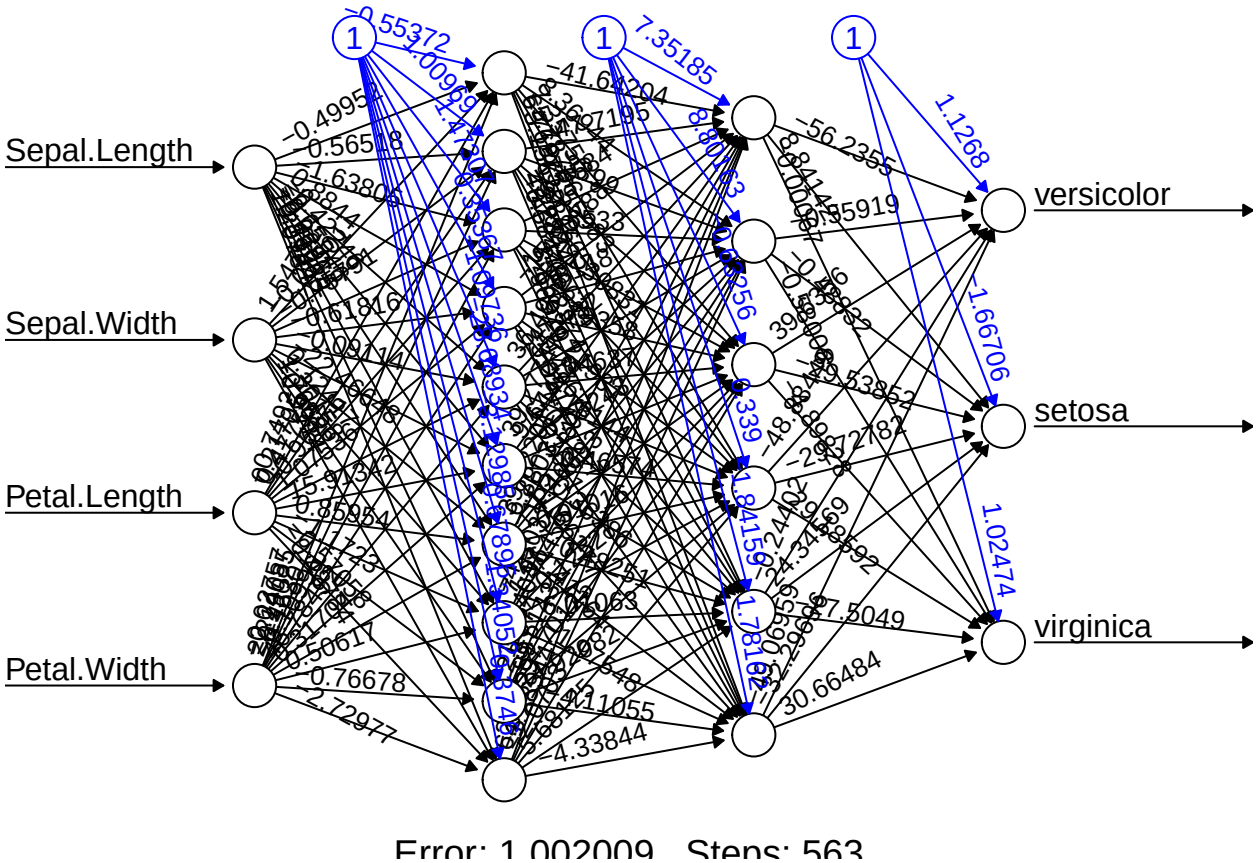
```
# Function to train and evaluate the model
train_and_evaluate <- function(hidden_layers) {
```

```
  model <- neuralnet(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
                     data = train_data, hidden = hidden_layers, linear.output = FALSE)
  plot(model, rep = 'best')

  # Model evaluation
  pred <- predict(model, test_data[, -5])
  labels <- c("setosa", "versicolor", "virginica")

  # Convert prediction to factor levels
  prediction_label <- tibble(max.col = max.col(pred)) %>%
    mutate(pred = labels[max.col]) %>%
    select(pred) %>%
    unlist()

  # Create confusion matrix
  confusion_matrix <- table(test_data$Species, prediction_label)
  print(confusion_matrix)

  # Calculate accuracy
  check <- as.numeric(test_data$Species) == max.col(pred)
  accuracy <- (sum(check) / nrow(test_data)) * 100
  print(paste("Accuracy for hidden layers", paste(hidden_layers, collapse = ","), ":", accuracy))

  # Print a few sample predictions
  sample_indices <- sample(seq_len(nrow(test_data)), 3)
  cat("Sample predictions:\n")
  for (i in sample_indices) {
    cat("Actual: ", as.character(test_data$Species[i]), " Predicted: ", as.character(prediction_label[i]
  }

  return(accuracy)
}

# Train and evaluate models with different hidden layers
hidden_layers_list <- list(c(4, 2), c(10, 6), c(112, 50), c(45, 30))
accuracies <- lapply(hidden_layers_list, train_and_evaluate)
```

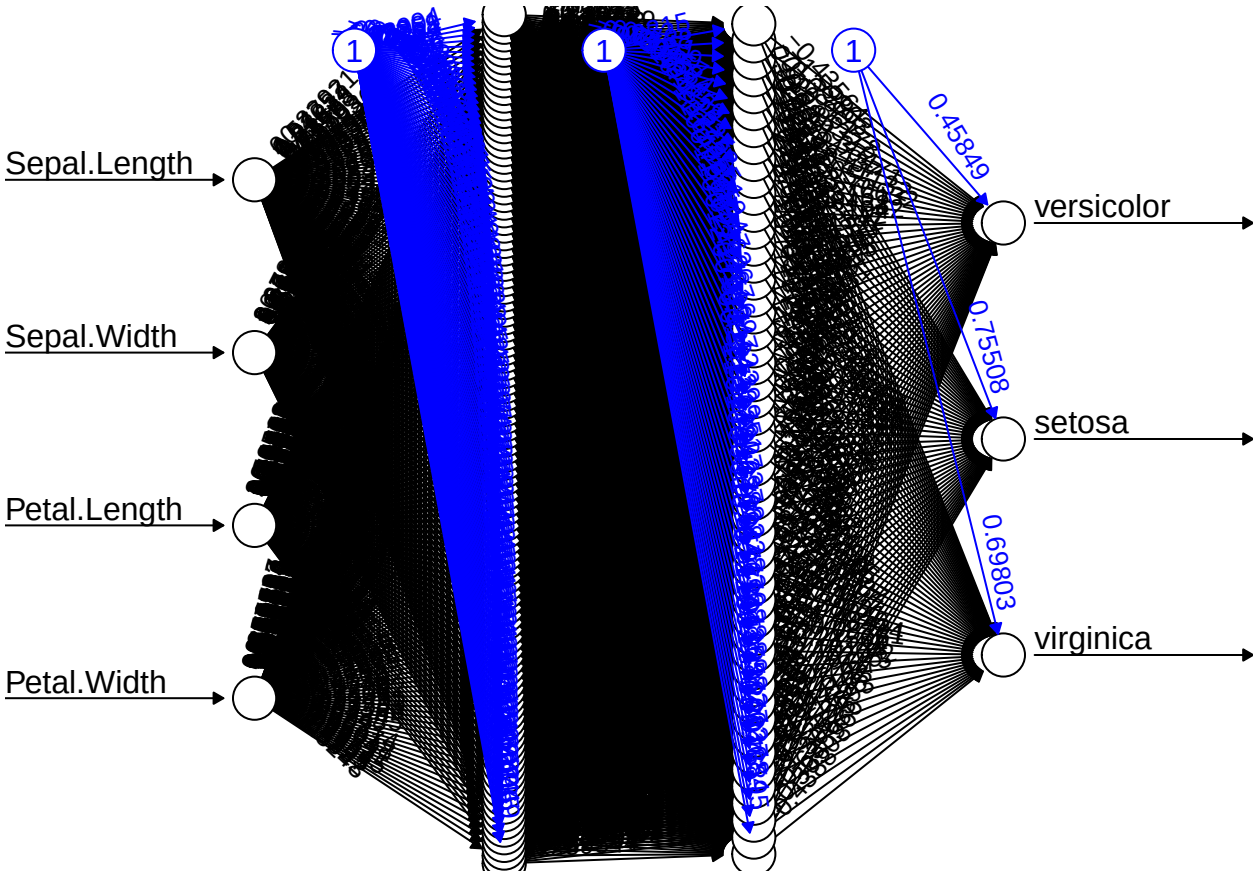


Error: 1.001034 Steps: 1058

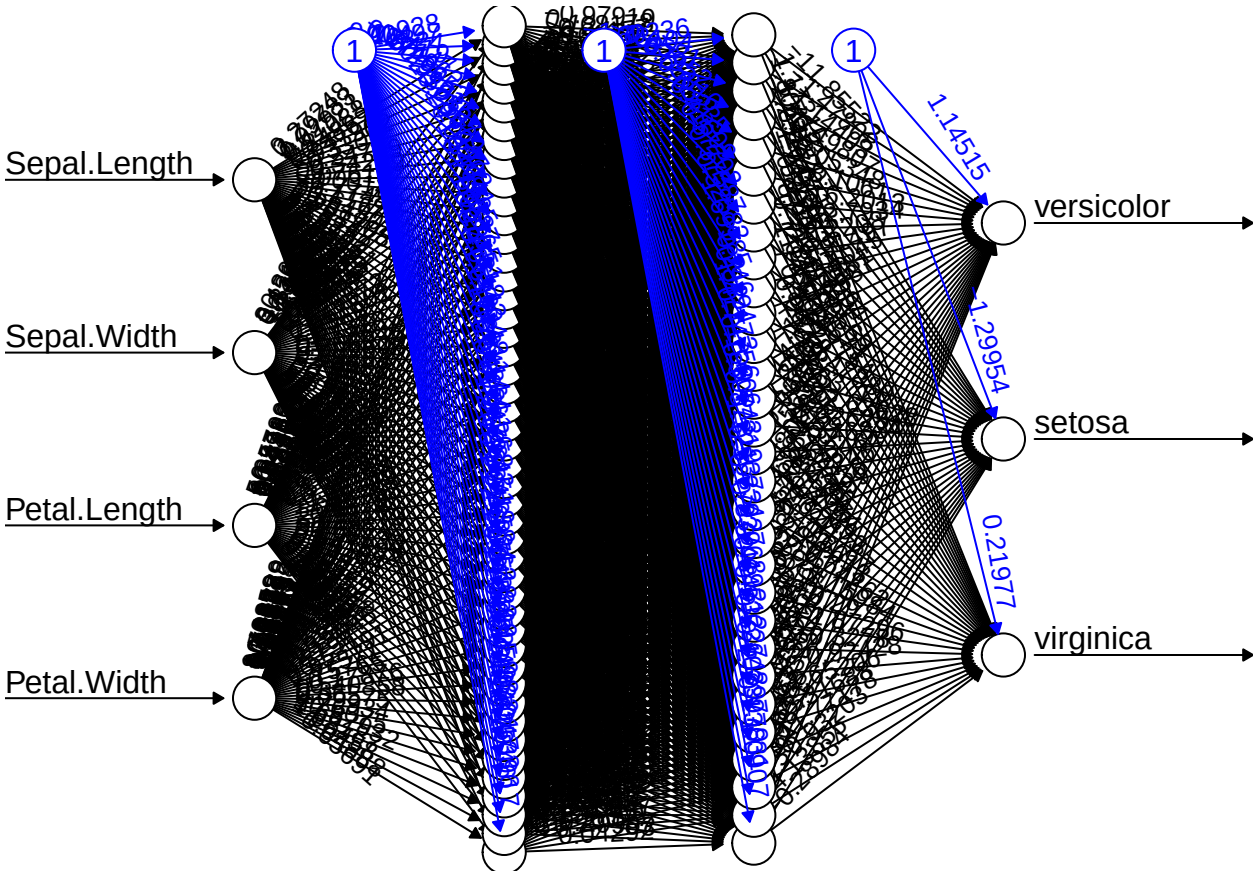
```
##             prediction_label
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0          9         0
##   virginica       0          0        11
## [1] "Accuracy for hidden layers 4,2 : 100"
## Sample predictions:
## Actual:  virginica  Predicted:  virginica
## Actual:  setosa  Predicted:  setosa
## Actual:  virginica  Predicted:  virginica
```




```
##             prediction_label
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0          9         0
##   virginica       0          0        11
## [1] "Accuracy for hidden layers 10,6 : 100"
## Sample predictions:
## Actual:  versicolor  Predicted:  versicolor
## Actual:  setosa  Predicted:  setosa
## Actual:  virginica  Predicted:  virginica
```




```
##             prediction_label
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0          9         0
##   virginica       0          0        11
## [1] "Accuracy for hidden layers 112,50 : 100"
## Sample predictions:
## Actual:  setosa  Predicted:  setosa
## Actual:  setosa  Predicted:  setosa
## Actual:  setosa  Predicted:  setosa
```




```
##             prediction_label
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0          8         1
##   virginica       0          0        11
## [1] "Accuracy for hidden layers 45,30 : 96.6666666666667"
## Sample predictions:
## Actual:  virginica  Predicted:  virginica
## Actual:  setosa  Predicted:  setosa
## Actual:  versicolor  Predicted:  versicolor
```

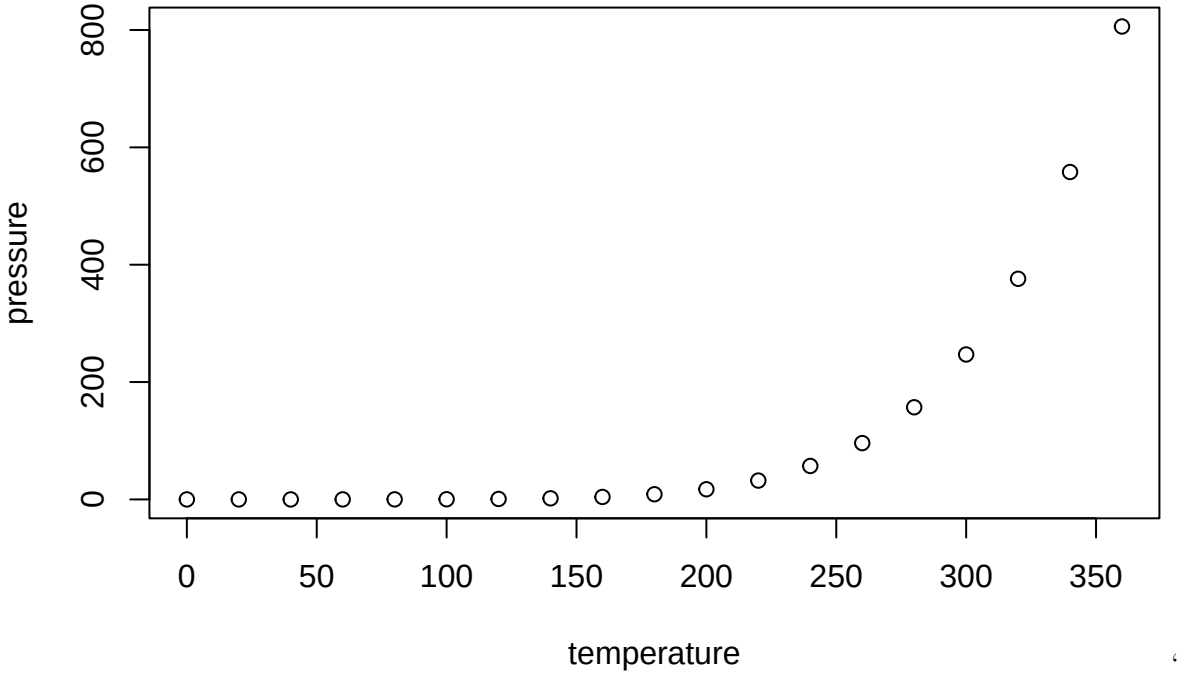
```r
# Print all accuracies
print(accuracies)
```

```
## [[1]]
## [1] 100
##
## [[2]]
## [1] 100
##
## [[3]]
## [1] 100
##
## [[4]]
## [1] 96.66667
```

Hidden Layer	Accuracy
4,2	100
10,6	100
112,50	100
45,30	96.67

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.