```
### Multiple Linear Regression Comparison with a Neural Network

# JSM 2018
# Poster 181 Classroom Demonstration: Deep Learning for Classification
#     and Regression, Introduction to GPU Computing"

# Eric A. Suess
# Department Of Statistics and Biostatistics
# CSU East Bay
# eric.suess@csueastbay.edu

# This example fits a multilayer NN to a numeric prediction problem.
# Mutliple Linear Regression could be used.
# h2o is used to demonstrate Deep Learning.

##### Lantz Machine Learning with R
##### Chapter 7: Neural Networks  -------------------

##### Part 1: Neural Networks -------------------
## Example: Modeling the Strength of Concrete  ----

## Step 2: Exploring and preparing the data ----
# read in data and examine structure
concrete <- read.csv("concrete.csv")
str(concrete)

# custom normalization function
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# apply normalization to entire data frame
concrete_norm <- as.data.frame(lapply(concrete, normalize))

# confirm that the range is now between zero and one
summary(concrete_norm$strength)

# compared to the original minimum and maximum
summary(concrete$strength)

# create training and test data
concrete_train <- concrete_norm[1:773, ]
concrete_test <- concrete_norm[774:1030, ]

## Step 3: Training a model on the data ----
# train the neuralnet model
library(neuralnet)

# simple ANN with only a single hidden neuron
set.seed(12345) # to guarantee repeatable results
concrete_model <- neuralnet(formula = strength ~ cement + slag +
                            ash + water + superplastic +
                            coarseagg + fineagg + age,
                            data = concrete_train)

# visualize the network topology
plot(concrete_model)

# Reference: http://www.r-bloggers.com/neuralnettools-1-0-0-now-on-cran/
# alternative plot
library(NeuralNetTools)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model, alpha = 0.6)

## Step 4: Evaluating model performance ----
# obtain model results
model_results <- compute(concrete_model, concrete_test[1:8])
# obtain predicted strength values
predicted_strength <- model_results$net.result
# examine the correlation between predicted and actual values
cor(predicted_strength, concrete_test$strength)   # higher than stated in book 0.7170368646

# produce actual predictions by

head(predicted_strength)

concrete_train_original_strength <- concrete[1:773,"strength"]

strength_min <- min(concrete_train_original_strength)
strength_max <- max(concrete_train_original_strength)

head(concrete_train_original_strength)

# custom normalization function
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}
```

```
### Multiple Linear Regression Comparison with a Neural Network

# JSM 2018
# Poster 181 Classroom Demonstration: Deep Learning for Classification
#     and Regression, Introduction to GPU Computing"

# Eric A. Suess
# Department Of Statistics and Biostatistics
# CSU East Bay
# eric.suess@csueastbay.edu

# This example fits a multilayer NN to a numeric prediction problem.
# Mutliple Linear Regression could be used.
# h2o is used to demonstrate Deep Learning.

##### Lantz Machine Learning with R
##### Chapter 7: Neural Networks  -------------------

##### Part 1: Neural Networks -------------------
## Example: Modeling the Strength of Concrete  ----

## Step 2: Exploring and preparing the data ----
# read in data and examine structure
concrete <- read.csv("concrete.csv")
str(concrete)

# custom normalization function
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# apply normalization to entire data frame
concrete_norm <- as.data.frame(lapply(concrete, normalize))

# confirm that the range is now between zero and one
summary(concrete_norm$strength)

# compared to the original minimum and maximum
summary(concrete$strength)

# create training and test data
concrete_train <- concrete_norm[1:773, ]
concrete_test <- concrete_norm[774:1030, ]

## Step 3: Training a model on the data ----
# train the neuralnet model
library(neuralnet)

# simple ANN with only a single hidden neuron
set.seed(12345) # to guarantee repeatable results
concrete_model <- neuralnet(formula = strength ~ cement + slag +
                            ash + water + superplastic +
                            coarseagg + fineagg + age,
                            data = concrete_train)

# visualize the network topology
plot(concrete_model)

# Reference: http://www.r-bloggers.com/neuralnettools-1-0-0-now-on-cran/
# alternative plot
library(NeuralNetTools)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model, alpha = 0.6)

## Step 4: Evaluating model performance ----
# obtain model results
model_results <- compute(concrete_model, concrete_test[1:8])
# obtain predicted strength values
predicted_strength <- model_results$net.result
# examine the correlation between predicted and actual values
cor(predicted_strength, concrete_test$strength)   # higher than stated in book 0.7170368646

# produce actual predictions by

head(predicted_strength)

concrete_train_original_strength <- concrete[1:773,"strength"]

strength_min <- min(concrete_train_original_strength)
strength_max <- max(concrete_train_original_strength)

head(concrete_train_original_strength)

# custom normalization function
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}
```

```
### Multiple Linear Regression Comparison with a Neural Network

# JSM 2018
# Poster 181 Classroom Demonstration: Deep Learning for Classification
#     and Regression, Introduction to GPU Computing"

# Eric A. Suess
# Department Of Statistics and Biostatistics
# CSU East Bay
# eric.suess@csueastbay.edu

# This example fits a multilayer NN to a numeric prediction problem.
# Mutliple Linear Regression could be used.
# h2o is used to demonstrate Deep Learning.

##### Lantz Machine Learning with R
##### Chapter 7: Neural Networks  -------------------

##### Part 1: Neural Networks -------------------
## Example: Modeling the Strength of Concrete  ----

## Step 2: Exploring and preparing the data ----
# read in data and examine structure
concrete <- read.csv("concrete.csv")
str(concrete)

# custom normalization function
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# apply normalization to entire data frame
concrete_norm <- as.data.frame(lapply(concrete, normalize))

# confirm that the range is now between zero and one
summary(concrete_norm$strength)

# compared to the original minimum and maximum
summary(concrete$strength)

# create training and test data
concrete_train <- concrete_norm[1:773, ]
concrete_test <- concrete_norm[774:1030, ]

## Step 3: Training a model on the data ----
# train the neuralnet model
library(neuralnet)

# simple ANN with only a single hidden neuron
set.seed(12345) # to guarantee repeatable results
concrete_model <- neuralnet(formula = strength ~ cement + slag +
                            ash + water + superplastic +
                            coarseagg + fineagg + age,
                            data = concrete_train)

# visualize the network topology
plot(concrete_model)

# Reference: http://www.r-bloggers.com/neuralnettools-1-0-0-now-on-cran/
# alternative plot
library(NeuralNetTools)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model, alpha = 0.6)

## Step 4: Evaluating model performance ----
# obtain model results
model_results <- compute(concrete_model, concrete_test[1:8])
# obtain predicted strength values
predicted_strength <- model_results$net.result
# examine the correlation between predicted and actual values
cor(predicted_strength, concrete_test$strength)   # higher than stated in book 0.7170368646

# produce actual predictions by

head(predicted_strength)

concrete_train_original_strength <- concrete[1:773,"strength"]

strength_min <- min(concrete_train_original_strength)
strength_max <- max(concrete_train_original_strength)

head(concrete_train_original_strength)

# custom normalization function
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}
```

```
### Multiple Linear Regression Comparison with a Neural Network

# JSM 2018
# Poster 181 Classroom Demonstration: Deep Learning for Classification
#     and Regression, Introduction to GPU Computing"

# Eric A. Suess
# Department Of Statistics and Biostatistics
# CSU East Bay
# eric.suess@csueastbay.edu

# This example fits a multilayer NN to a numeric prediction problem.
# Mutliple Linear Regression could be used.
# h2o is used to demonstrate Deep Learning.

##### Lantz Machine Learning with R
##### Chapter 7: Neural Networks  -------------------

##### Part 1: Neural Networks -------------------
## Example: Modeling the Strength of Concrete  ----

## Step 2: Exploring and preparing the data ----
# read in data and examine structure
concrete <- read.csv("concrete.csv")
str(concrete)

# custom normalization function
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# apply normalization to entire data frame
concrete_norm <- as.data.frame(lapply(concrete, normalize))

# confirm that the range is now between zero and one
summary(concrete_norm$strength)

# compared to the original minimum and maximum
summary(concrete$strength)

# create training and test data
concrete_train <- concrete_norm[1:773, ]
concrete_test <- concrete_norm[774:1030, ]

## Step 3: Training a model on the data ----
# train the neuralnet model
library(neuralnet)

# simple ANN with only a single hidden neuron
set.seed(12345) # to guarantee repeatable results
concrete_model <- neuralnet(formula = strength ~ cement + slag +
                            ash + water + superplastic +
                            coarseagg + fineagg + age,
                            data = concrete_train)

# visualize the network topology
plot(concrete_model)

# Reference: http://www.r-bloggers.com/neuralnettools-1-0-0-now-on-cran/
# alternative plot
library(NeuralNetTools)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model, alpha = 0.6)

## Step 4: Evaluating model performance ----
# obtain model results
model_results <- compute(concrete_model, concrete_test[1:8])
# obtain predicted strength values
predicted_strength <- model_results$net.result
# examine the correlation between predicted and actual values
cor(predicted_strength, concrete_test$strength)   # higher than stated in book 0.7170368646

# produce actual predictions by

head(predicted_strength)

concrete_train_original_strength <- concrete[1:773,"strength"]

strength_min <- min(concrete_train_original_strength)
strength_max <- max(concrete_train_original_strength)

head(concrete_train_original_strength)

# custom normalization function
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}
```

```
### Multiple Linear Regression Comparison with a Neural Network

# JSM 2018
# Poster 181 Classroom Demonstration: Deep Learning for Classification
#     and Regression, Introduction to GPU Computing"

# Eric A. Suess
# Department Of Statistics and Biostatistics
# CSU East Bay
# eric.suess@csueastbay.edu

# This example fits a multilayer NN to a numeric prediction problem.
# Mutliple Linear Regression could be used.
# h2o is used to demonstrate Deep Learning.

##### Lantz Machine Learning with R
##### Chapter 7: Neural Networks  -------------------

##### Part 1: Neural Networks -------------------
## Example: Modeling the Strength of Concrete  ----

## Step 2: Exploring and preparing the data ----
# read in data and examine structure
concrete <- read.csv("concrete.csv")
str(concrete)

# custom normalization function
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# apply normalization to entire data frame
concrete_norm <- as.data.frame(lapply(concrete, normalize))

# confirm that the range is now between zero and one
summary(concrete_norm$strength)

# compared to the original minimum and maximum
summary(concrete$strength)

# create training and test data
concrete_train <- concrete_norm[1:773, ]
concrete_test <- concrete_norm[774:1030, ]

## Step 3: Training a model on the data ----
# train the neuralnet model
library(neuralnet)

# simple ANN with only a single hidden neuron
set.seed(12345) # to guarantee repeatable results
concrete_model <- neuralnet(formula = strength ~ cement + slag +
                            ash + water + superplastic +
                            coarseagg + fineagg + age,
                            data = concrete_train)

# visualize the network topology
plot(concrete_model)

# Reference: http://www.r-bloggers.com/neuralnettools-1-0-0-now-on-cran/
# alternative plot
library(NeuralNetTools)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model, alpha = 0.6)

## Step 4: Evaluating model performance ----
# obtain model results
model_results <- compute(concrete_model, concrete_test[1:8])
# obtain predicted strength values
predicted_strength <- model_results$net.result
# examine the correlation between predicted and actual values
cor(predicted_strength, concrete_test$strength)   # higher than stated in book 0.7170368646

# produce actual predictions by

head(predicted_strength)

concrete_train_original_strength <- concrete[1:773,"strength"]

strength_min <- min(concrete_train_original_strength)
strength_max <- max(concrete_train_original_strength)

head(concrete_train_original_strength)

# custom normalization function
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}
```

```
### Multiple Linear Regression Comparison with a Neural Network

# JSM 2018
# Poster 181 Classroom Demonstration: Deep Learning for Classification
#     and Regression, Introduction to GPU Computing"

# Eric A. Suess
# Department Of Statistics and Biostatistics
# CSU East Bay
# eric.suess@csueastbay.edu

# This example fits a multilayer NN to a numeric prediction problem.
# Mutliple Linear Regression could be used.
# h2o is used to demonstrate Deep Learning.

##### Lantz Machine Learning with R
##### Chapter 7: Neural Networks  -------------------

##### Part 1: Neural Networks -------------------
## Example: Modeling the Strength of Concrete  ----

## Step 2: Exploring and preparing the data ----
# read in data and examine structure
concrete <- read.csv("concrete.csv")
str(concrete)

# custom normalization function
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# apply normalization to entire data frame
concrete_norm <- as.data.frame(lapply(concrete, normalize))

# confirm that the range is now between zero and one
summary(concrete_norm$strength)

# compared to the original minimum and maximum
summary(concrete$strength)

# create training and test data
concrete_train <- concrete_norm[1:773, ]
concrete_test <- concrete_norm[774:1030, ]

## Step 3: Training a model on the data ----
# train the neuralnet model
library(neuralnet)

# simple ANN with only a single hidden neuron
set.seed(12345) # to guarantee repeatable results
concrete_model <- neuralnet(formula = strength ~ cement + slag +
                            ash + water + superplastic +
                            coarseagg + fineagg + age,
                            data = concrete_train)

# visualize the network topology
plot(concrete_model)

# Reference: http://www.r-bloggers.com/neuralnettools-1-0-0-now-on-cran/
# alternative plot
library(NeuralNetTools)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model, alpha = 0.6)

## Step 4: Evaluating model performance ----
# obtain model results
model_results <- compute(concrete_model, concrete_test[1:8])
# obtain predicted strength values
predicted_strength <- model_results$net.result
# examine the correlation between predicted and actual values
cor(predicted_strength, concrete_test$strength)   # higher than stated in book 0.7170368646

# produce actual predictions by

head(predicted_strength)

concrete_train_original_strength <- concrete[1:773,"strength"]

strength_min <- min(concrete_train_original_strength)
strength_max <- max(concrete_train_original_strength)

head(concrete_train_original_strength)

# custom normalization function
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}
```

```
### Multiple Linear Regression Comparison with a Neural Network

# JSM 2018
# Poster 181 Classroom Demonstration: Deep Learning for Classification
#     and Regression, Introduction to GPU Computing"

# Eric A. Suess
# Department Of Statistics and Biostatistics
# CSU East Bay
# eric.suess@csueastbay.edu

# This example fits a multilayer NN to a numeric prediction problem.
# Mutliple Linear Regression could be used.
# h2o is used to demonstrate Deep Learning.

##### Lantz Machine Learning with R
##### Chapter 7: Neural Networks  -------------------

##### Part 1: Neural Networks -------------------
## Example: Modeling the Strength of Concrete  ----

## Step 2: Exploring and preparing the data ----
# read in data and examine structure
concrete <- read.csv("concrete.csv")
str(concrete)

# custom normalization function
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# apply normalization to entire data frame
concrete_norm <- as.data.frame(lapply(concrete, normalize))

# confirm that the range is now between zero and one
summary(concrete_norm$strength)

# compared to the original minimum and maximum
summary(concrete$strength)

# create training and test data
concrete_train <- concrete_norm[1:773, ]
concrete_test <- concrete_norm[774:1030, ]

## Step 3: Training a model on the data ----
# train the neuralnet model
library(neuralnet)

# simple ANN with only a single hidden neuron
set.seed(12345) # to guarantee repeatable results
concrete_model <- neuralnet(formula = strength ~ cement + slag +
                            ash + water + superplastic +
                            coarseagg + fineagg + age,
                            data = concrete_train)

# visualize the network topology
plot(concrete_model)

# Reference: http://www.r-bloggers.com/neuralnettools-1-0-0-now-on-cran/
# alternative plot
library(NeuralNetTools)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model, alpha = 0.6)

## Step 4: Evaluating model performance ----
# obtain model results
model_results <- compute(concrete_model, concrete_test[1:8])
# obtain predicted strength values
predicted_strength <- model_results$net.result
# examine the correlation between predicted and actual values
cor(predicted_strength, concrete_test$strength)   # higher than stated in book 0.7170368646

# produce actual predictions by

head(predicted_strength)

concrete_train_original_strength <- concrete[1:773,"strength"]

strength_min <- min(concrete_train_original_strength)
strength_max <- max(concrete_train_original_strength)

head(concrete_train_original_strength)

# custom normalization function
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}
```

```
### Multiple Linear Regression Comparison with a Neural Network

# JSM 2018
# Poster 181 Classroom Demonstration: Deep Learning for Classification
#     and Regression, Introduction to GPU Computing"

# Eric A. Suess
# Department Of Statistics and Biostatistics
# CSU East Bay
# eric.suess@csueastbay.edu

# This example fits a multilayer NN to a numeric prediction problem.
# Mutliple Linear Regression could be used.
# h2o is used to demonstrate Deep Learning.

##### Lantz Machine Learning with R
##### Chapter 7: Neural Networks  -------------------

##### Part 1: Neural Networks -------------------
## Example: Modeling the Strength of Concrete  ----

## Step 2: Exploring and preparing the data ----
# read in data and examine structure
concrete <- read.csv("concrete.csv")
str(concrete)

# custom normalization function
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# apply normalization to entire data frame
concrete_norm <- as.data.frame(lapply(concrete, normalize))

# confirm that the range is now between zero and one
summary(concrete_norm$strength)

# compared to the original minimum and maximum
summary(concrete$strength)

# create training and test data
concrete_train <- concrete_norm[1:773, ]
concrete_test <- concrete_norm[774:1030, ]

## Step 3: Training a model on the data ----
# train the neuralnet model
library(neuralnet)

# simple ANN with only a single hidden neuron
set.seed(12345) # to guarantee repeatable results
concrete_model <- neuralnet(formula = strength ~ cement + slag +
                            ash + water + superplastic +
                            coarseagg + fineagg + age,
                            data = concrete_train)

# visualize the network topology
plot(concrete_model)

# Reference: http://www.r-bloggers.com/neuralnettools-1-0-0-now-on-cran/
# alternative plot
library(NeuralNetTools)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model, alpha = 0.6)

## Step 4: Evaluating model performance ----
# obtain model results
model_results <- compute(concrete_model, concrete_test[1:8])
# obtain predicted strength values
predicted_strength <- model_results$net.result
# examine the correlation between predicted and actual values
cor(predicted_strength, concrete_test$strength)   # higher than stated in book 0.7170368646

# produce actual predictions by

head(predicted_strength)

concrete_train_original_strength <- concrete[1:773,"strength"]

strength_min <- min(concrete_train_original_strength)
strength_max <- max(concrete_train_original_strength)

head(concrete_train_original_strength)

# custom normalization function
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}
```

```
### Multiple Linear Regression Comparison with a Neural Network

# JSM 2018
# Poster 181 Classroom Demonstration: Deep Learning for Classification
#     and Regression, Introduction to GPU Computing"

# Eric A. Suess
# Department Of Statistics and Biostatistics
# CSU East Bay
# eric.suess@csueastbay.edu

# This example fits a multilayer NN to a numeric prediction problem.
# Mutliple Linear Regression could be used.
# h2o is used to demonstrate Deep Learning.

##### Lantz Machine Learning with R
##### Chapter 7: Neural Networks  -------------------

##### Part 1: Neural Networks -------------------
## Example: Modeling the Strength of Concrete  ----

## Step 2: Exploring and preparing the data ----
# read in data and examine structure
concrete <- read.csv("concrete.csv")
str(concrete)

# custom normalization function
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# apply normalization to entire data frame
concrete_norm <- as.data.frame(lapply(concrete, normalize))

# confirm that the range is now between zero and one
summary(concrete_norm$strength)

# compared to the original minimum and maximum
summary(concrete$strength)

# create training and test data
concrete_train <- concrete_norm[1:773, ]
concrete_test <- concrete_norm[774:1030, ]

## Step 3: Training a model on the data ----
# train the neuralnet model
library(neuralnet)

# simple ANN with only a single hidden neuron
set.seed(12345) # to guarantee repeatable results
concrete_model <- neuralnet(formula = strength ~ cement + slag +
                            ash + water + superplastic +
                            coarseagg + fineagg + age,
                            data = concrete_train)

# visualize the network topology
plot(concrete_model)

# Reference: http://www.r-bloggers.com/neuralnettools-1-0-0-now-on-cran/
# alternative plot
library(NeuralNetTools)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model, alpha = 0.6)

## Step 4: Evaluating model performance ----
# obtain model results
model_results <- compute(concrete_model, concrete_test[1:8])
# obtain predicted strength values
predicted_strength <- model_results$net.result
# examine the correlation between predicted and actual values
cor(predicted_strength, concrete_test$strength)   # higher than stated in book 0.7170368646

# produce actual predictions by

head(predicted_strength)

concrete_train_original_strength <- concrete[1:773,"strength"]

strength_min <- min(concrete_train_original_strength)
strength_max <- max(concrete_train_original_strength)

head(concrete_train_original_strength)

# custom normalization function
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}
```

```
### Multiple Linear Regression Comparison with a Neural Network

# JSM 2018
# Poster 181 Classroom Demonstration: Deep Learning for Classification
#     and Regression, Introduction to GPU Computing"

# Eric A. Suess
# Department Of Statistics and Biostatistics
# CSU East Bay
# eric.suess@csueastbay.edu

# This example fits a multilayer NN to a numeric prediction problem.
# Mutliple Linear Regression could be used.
# h2o is used to demonstrate Deep Learning.

##### Lantz Machine Learning with R
##### Chapter 7: Neural Networks  -------------------

##### Part 1: Neural Networks -------------------
## Example: Modeling the Strength of Concrete  ----

## Step 2: Exploring and preparing the data ----
# read in data and examine structure
concrete <- read.csv("concrete.csv")
str(concrete)

# custom normalization function
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# apply normalization to entire data frame
concrete_norm <- as.data.frame(lapply(concrete, normalize))

# confirm that the range is now between zero and one
summary(concrete_norm$strength)

# compared to the original minimum and maximum
summary(concrete$strength)

# create training and test data
concrete_train <- concrete_norm[1:773, ]
concrete_test <- concrete_norm[774:1030, ]

## Step 3: Training a model on the data ----
# train the neuralnet model
library(neuralnet)

# simple ANN with only a single hidden neuron
set.seed(12345) # to guarantee repeatable results
concrete_model <- neuralnet(formula = strength ~ cement + slag +
                            ash + water + superplastic +
                            coarseagg + fineagg + age,
                            data = concrete_train)

# visualize the network topology
plot(concrete_model)

# Reference: http://www.r-bloggers.com/neuralnettools-1-0-0-now-on-cran/
# alternative plot
library(NeuralNetTools)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model, alpha = 0.6)

## Step 4: Evaluating model performance ----
# obtain model results
model_results <- compute(concrete_model, concrete_test[1:8])
# obtain predicted strength values
predicted_strength <- model_results$net.result
# examine the correlation between predicted and actual values
cor(predicted_strength, concrete_test$strength)   # higher than stated in book 0.7170368646

# produce actual predictions by

head(predicted_strength)

concrete_train_original_strength <- concrete[1:773,"strength"]

strength_min <- min(concrete_train_original_strength)
strength_max <- max(concrete_train_original_strength)

head(concrete_train_original_strength)

# custom normalization function
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}
```

```
strength_pred <- unnormalize(predicted_strength, strength_min, strength_max)
strength_pred

## Step 5: Improving model performance ----
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "logistic")

# plot the network
plot(concrete_model2)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model2, alpha = 0.6)

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)  # higher than stated in book 0.801444583

# try different activation function
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "tanh")

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)

# using h2o deeplearning

library(h2o)

h2o.init(nthreads=8, max_mem_size="2G")
h2o.removeAll() ## clean slate - just in case the cluster was already running

h2o.init()

concrete.hex <- h2o.importFile("concrete.csv")

summary(concrete.hex)

splits <- h2o.splitFrame(concrete.hex, 0.75, seed=1234)

dl <- h2o.deeplearning(x=1:8,y="strength",training_frame=splits[[1]],activation = "Tanh",
                       hidden = c(200,200), distribution = "gaussian")

dl.predict <- h2o.predict(dl, splits[[2]])

cor(as.vector(dl.predict), as.vector(splits[[2]]$strength))

dl@parameters

h2o.performance(dl)

h2o.shutdown()
```

```
strength_pred <- unnormalize(predicted_strength, strength_min, strength_max)
strength_pred

## Step 5: Improving model performance ----
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "logistic")

# plot the network
plot(concrete_model2)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model2, alpha = 0.6)

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)  # higher than stated in book 0.801444583

# try different activation function
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "tanh")

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)

# using h2o deeplearning

library(h2o)

h2o.init(nthreads=8, max_mem_size="2G")
h2o.removeAll() ## clean slate - just in case the cluster was already running

h2o.init()

concrete.hex <- h2o.importFile("concrete.csv")

summary(concrete.hex)

splits <- h2o.splitFrame(concrete.hex, 0.75, seed=1234)

dl <- h2o.deeplearning(x=1:8,y="strength",training_frame=splits[[1]],activation = "Tanh",
                       hidden = c(200,200), distribution = "gaussian")

dl.predict <- h2o.predict(dl, splits[[2]])

cor(as.vector(dl.predict), as.vector(splits[[2]]$strength))

dl@parameters

h2o.performance(dl)

h2o.shutdown()
```

```
strength_pred <- unnormalize(predicted_strength, strength_min, strength_max)
strength_pred

## Step 5: Improving model performance ----
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "logistic")

# plot the network
plot(concrete_model2)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model2, alpha = 0.6)

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)  # higher than stated in book 0.801444583

# try different activation function
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "tanh")

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)

# using h2o deeplearning

library(h2o)

h2o.init(nthreads=8, max_mem_size="2G")
h2o.removeAll() ## clean slate - just in case the cluster was already running

h2o.init()

concrete.hex <- h2o.importFile("concrete.csv")

summary(concrete.hex)

splits <- h2o.splitFrame(concrete.hex, 0.75, seed=1234)

dl <- h2o.deeplearning(x=1:8,y="strength",training_frame=splits[[1]],activation = "Tanh",
                       hidden = c(200,200), distribution = "gaussian")

dl.predict <- h2o.predict(dl, splits[[2]])

cor(as.vector(dl.predict), as.vector(splits[[2]]$strength))

dl@parameters

h2o.performance(dl)

h2o.shutdown()
```

```
strength_pred <- unnormalize(predicted_strength, strength_min, strength_max)
strength_pred

## Step 5: Improving model performance ----
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "logistic")

# plot the network
plot(concrete_model2)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model2, alpha = 0.6)

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)  # higher than stated in book 0.801444583

# try different activation function
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "tanh")

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)

# using h2o deeplearning

library(h2o)

h2o.init(nthreads=8, max_mem_size="2G")
h2o.removeAll() ## clean slate - just in case the cluster was already running

h2o.init()

concrete.hex <- h2o.importFile("concrete.csv")

summary(concrete.hex)

splits <- h2o.splitFrame(concrete.hex, 0.75, seed=1234)

dl <- h2o.deeplearning(x=1:8,y="strength",training_frame=splits[[1]],activation = "Tanh",
                       hidden = c(200,200), distribution = "gaussian")

dl.predict <- h2o.predict(dl, splits[[2]])

cor(as.vector(dl.predict), as.vector(splits[[2]]$strength))

dl@parameters

h2o.performance(dl)

h2o.shutdown()
```

```
strength_pred <- unnormalize(predicted_strength, strength_min, strength_max)
strength_pred

## Step 5: Improving model performance ----
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "logistic")

# plot the network
plot(concrete_model2)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model2, alpha = 0.6)

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)  # higher than stated in book 0.801444583

# try different activation function
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "tanh")

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)

# using h2o deeplearning

library(h2o)

h2o.init(nthreads=8, max_mem_size="2G")
h2o.removeAll() ## clean slate - just in case the cluster was already running

h2o.init()

concrete.hex <- h2o.importFile("concrete.csv")

summary(concrete.hex)

splits <- h2o.splitFrame(concrete.hex, 0.75, seed=1234)

dl <- h2o.deeplearning(x=1:8,y="strength",training_frame=splits[[1]],activation = "Tanh",
                       hidden = c(200,200), distribution = "gaussian")

dl.predict <- h2o.predict(dl, splits[[2]])

cor(as.vector(dl.predict), as.vector(splits[[2]]$strength))

dl@parameters

h2o.performance(dl)

h2o.shutdown()
```

```
strength_pred <- unnormalize(predicted_strength, strength_min, strength_max)
strength_pred

## Step 5: Improving model performance ----
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "logistic")

# plot the network
plot(concrete_model2)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model2, alpha = 0.6)

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)  # higher than stated in book 0.801444583

# try different activation function
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "tanh")

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)

# using h2o deeplearning

library(h2o)

h2o.init(nthreads=8, max_mem_size="2G")
h2o.removeAll() ## clean slate - just in case the cluster was already running

h2o.init()

concrete.hex <- h2o.importFile("concrete.csv")

summary(concrete.hex)

splits <- h2o.splitFrame(concrete.hex, 0.75, seed=1234)

dl <- h2o.deeplearning(x=1:8,y="strength",training_frame=splits[[1]],activation = "Tanh",
                       hidden = c(200,200), distribution = "gaussian")

dl.predict <- h2o.predict(dl, splits[[2]])

cor(as.vector(dl.predict), as.vector(splits[[2]]$strength))

dl@parameters

h2o.performance(dl)

h2o.shutdown()
```

```
strength_pred <- unnormalize(predicted_strength, strength_min, strength_max)
strength_pred

## Step 5: Improving model performance ----
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "logistic")

# plot the network
plot(concrete_model2)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model2, alpha = 0.6)

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)  # higher than stated in book 0.801444583

# try different activation function
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "tanh")

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)

# using h2o deeplearning

library(h2o)

h2o.init(nthreads=8, max_mem_size="2G")
h2o.removeAll() ## clean slate - just in case the cluster was already running

h2o.init()

concrete.hex <- h2o.importFile("concrete.csv")

summary(concrete.hex)

splits <- h2o.splitFrame(concrete.hex, 0.75, seed=1234)

dl <- h2o.deeplearning(x=1:8,y="strength",training_frame=splits[[1]],activation = "Tanh",
                       hidden = c(200,200), distribution = "gaussian")

dl.predict <- h2o.predict(dl, splits[[2]])

cor(as.vector(dl.predict), as.vector(splits[[2]]$strength))

dl@parameters

h2o.performance(dl)

h2o.shutdown()
```

```
strength_pred <- unnormalize(predicted_strength, strength_min, strength_max)
strength_pred

## Step 5: Improving model performance ----
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "logistic")

# plot the network
plot(concrete_model2)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model2, alpha = 0.6)

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)  # higher than stated in book 0.801444583

# try different activation function
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "tanh")

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)

# using h2o deeplearning

library(h2o)

h2o.init(nthreads=8, max_mem_size="2G")
h2o.removeAll() ## clean slate - just in case the cluster was already running

h2o.init()

concrete.hex <- h2o.importFile("concrete.csv")

summary(concrete.hex)

splits <- h2o.splitFrame(concrete.hex, 0.75, seed=1234)

dl <- h2o.deeplearning(x=1:8,y="strength",training_frame=splits[[1]],activation = "Tanh",
                       hidden = c(200,200), distribution = "gaussian")

dl.predict <- h2o.predict(dl, splits[[2]])

cor(as.vector(dl.predict), as.vector(splits[[2]]$strength))

dl@parameters

h2o.performance(dl)

h2o.shutdown()
```

```
strength_pred <- unnormalize(predicted_strength, strength_min, strength_max)
strength_pred

## Step 5: Improving model performance ----
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "logistic")

# plot the network
plot(concrete_model2)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model2, alpha = 0.6)

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)  # higher than stated in book 0.801444583

# try different activation function
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "tanh")

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)

# using h2o deeplearning

library(h2o)

h2o.init(nthreads=8, max_mem_size="2G")
h2o.removeAll() ## clean slate - just in case the cluster was already running

h2o.init()

concrete.hex <- h2o.importFile("concrete.csv")

summary(concrete.hex)

splits <- h2o.splitFrame(concrete.hex, 0.75, seed=1234)

dl <- h2o.deeplearning(x=1:8,y="strength",training_frame=splits[[1]],activation = "Tanh",
                       hidden = c(200,200), distribution = "gaussian")

dl.predict <- h2o.predict(dl, splits[[2]])

cor(as.vector(dl.predict), as.vector(splits[[2]]$strength))

dl@parameters

h2o.performance(dl)

h2o.shutdown()
```

```
strength_pred <- unnormalize(predicted_strength, strength_min, strength_max)
strength_pred

## Step 5: Improving model performance ----
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "logistic")

# plot the network
plot(concrete_model2)

# plotnet
par(mar = numeric(4), family = 'serif')
plotnet(concrete_model2, alpha = 0.6)

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)  # higher than stated in book 0.801444583

# try different activation function
# a more complex neural network topology with 5 hidden neurons
set.seed(12345) # to guarantee repeatable results
concrete_model2 <- neuralnet(strength ~ cement + slag +
                             ash + water + superplastic +
                             coarseagg + fineagg + age,
                             data = concrete_train, hidden = 5, act.fct = "tanh")

# evaluate the results as we did before
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)

# using h2o deeplearning

library(h2o)

h2o.init(nthreads=8, max_mem_size="2G")
h2o.removeAll() ## clean slate - just in case the cluster was already running

h2o.init()

concrete.hex <- h2o.importFile("concrete.csv")

summary(concrete.hex)

splits <- h2o.splitFrame(concrete.hex, 0.75, seed=1234)

dl <- h2o.deeplearning(x=1:8,y="strength",training_frame=splits[[1]],activation = "Tanh",
                       hidden = c(200,200), distribution = "gaussian")

dl.predict <- h2o.predict(dl, splits[[2]])

cor(as.vector(dl.predict), as.vector(splits[[2]]$strength))

dl@parameters

h2o.performance(dl)

h2o.shutdown()
```