Neural Networks, Regression Task, Exercise.

name

xx/xx/xxxx

1. Divide the data into 75% training and 25% testing & normalize the data

```r
if (!requireNamespace("tidyverse")) install.packages('tidyverse')
```

```
## Loading required namespace: tidyverse
```

```r
if (!requireNamespace("caret")) install.packages('caret')
```

```
## Loading required namespace: caret
```

```r
if (!requireNamespace("neuralnet")) install.packages('neuralnet')
```

```
## Loading required namespace: neuralnet
```

```r
library(tidyverse)
```

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

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

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

```r
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(neuralnet)
```

```
##
## Attaching package: 'neuralnet'

## The following object is masked from 'package:dplyr':
##
##     compute
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
data("Boston")
data = Boston
data <- subset(data, select = -c(rad))
# mean & standard deviation of the response
mean = mean(data$medv)
sd = sd(data$medv)
# normalize the data
data = data.frame(scale(data))

set.seed(123)
training.samples <- data$medv %>%
  createDataPartition(p = 0.75, list = FALSE)
train.data  <- data[training.samples, ]
test.data <- data[-training.samples, ]
str(train.data) # 381 obs
```

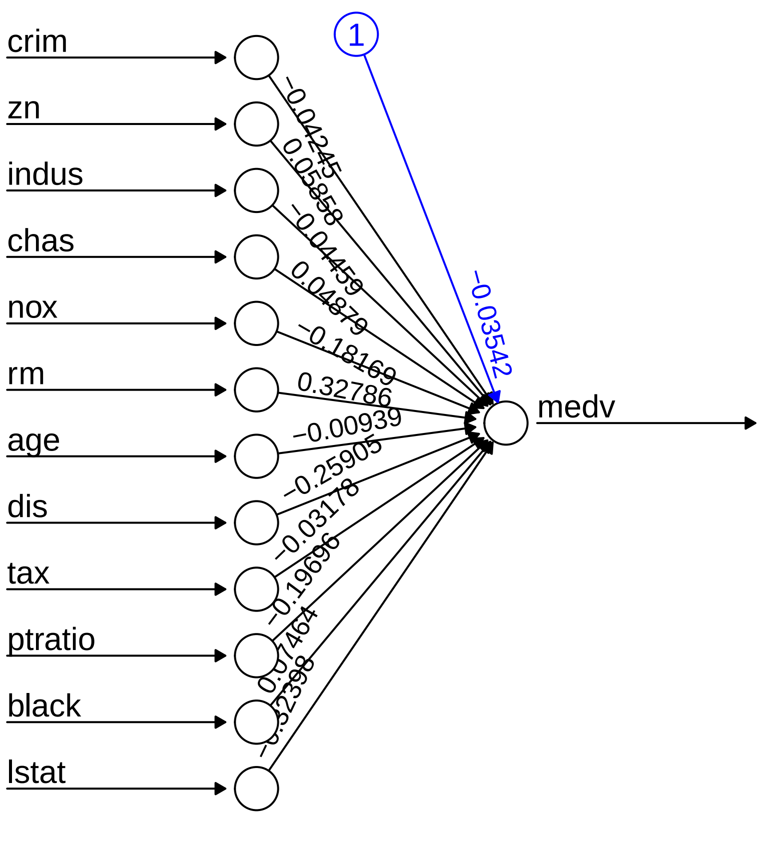
```
## 'data.frame':    381 obs. of  13 variables:
##  $ crim   : num  -0.419 -0.417 -0.416 -0.412 -0.41 ...
##  $ zn     : num  0.2845 -0.4872 -0.4872 -0.4872 0.0487 ...
##  $ indus  : num  -1.287 -0.593 -1.306 -1.306 -0.476 ...
##  $ chas   : num  -0.272 -0.272 -0.272 -0.272 -0.272 ...
##  $ nox    : num  -0.144 -0.74 -0.834 -0.834 -0.265 ...
##  $ rm     : num  0.413 0.194 1.015 1.227 -0.388 ...
##  $ age    : num  -0.1199 0.3668 -0.8091 -0.5107 -0.0702 ...
##  $ dis    : num  0.14 0.557 1.077 1.077 0.838 ...
##  $ tax    : num  -0.666 -0.986 -1.105 -1.105 -0.577 ...
##  $ ptratio: num  -1.458 -0.303 0.113 0.113 -1.504 ...
##  $ black  : num  0.441 0.441 0.416 0.441 0.426 ...
##  $ lstat  : num  -1.0745 -0.492 -1.3602 -1.0255 -0.0312 ...
##  $ medv   : num  0.1595 -0.1014 1.1816 1.486 0.0399 ...
```

```r
str(test.data) # 125 obs
```

```
## 'data.frame':    125 obs. of  13 variables:
##  $ crim   : num  -0.417 -0.417 -0.396 -0.394 -0.347 ...
##  $ zn     : num  -0.4872 -0.4872 0.0487 0.0487 -0.4872 ...
##  $ indus  : num  -0.593 -1.306 -0.476 -0.476 -0.437 ...
##  $ chas   : num  -0.272 -0.272 -0.272 -0.272 -0.272 ...
##  $ nox    : num  -0.74 -0.834 -0.265 -0.265 -0.144 ...
##  $ rm     : num  1.281 0.207 -0.93 0.131 -0.478 ...
##  $ age    : num  -0.266 -0.351 1.116 0.914 -0.241 ...
##  $ dis    : num  0.557 1.077 1.086 1.212 0.433 ...
##  $ tax    : num  -0.986 -1.105 -0.577 -0.577 -0.601 ...
##  $ ptratio: num  -0.303 0.113 -1.504 -1.504 1.175 ...
##  $ black  : num  0.396 0.41 0.328 0.393 0.441 ...
##  $ lstat  : num  -1.208 -1.042 2.419 1.092 -0.615 ...
##  $ medv   : num  1.323 0.671 -0.656 -0.819 -0.232 ...
```

2. NN model with (i) no hidden layer, (ii) the default loss function of 'sse', and (iii) the default activation function of 'identity'.

```r
set.seed(123)
nn = neuralnet(medv~., data = train.data, hidden = 0, err.fct = "sse", linear.output = T)
plot(nn, rep = 'best')
```

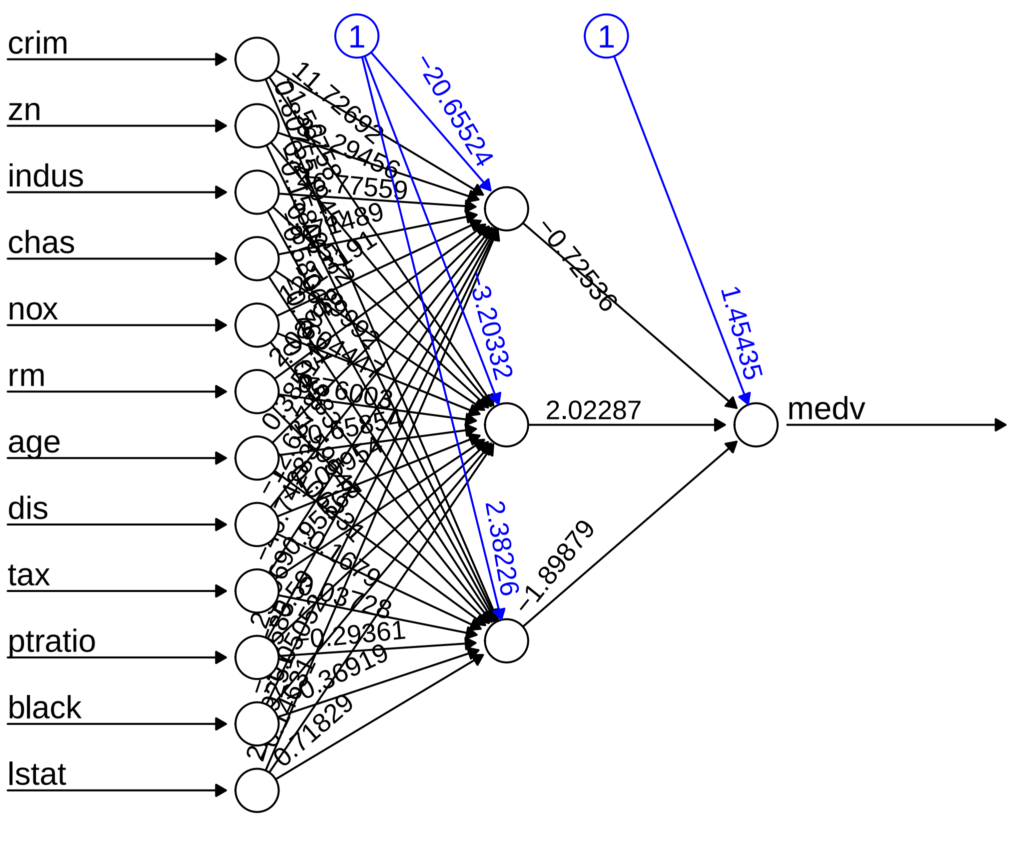


```
pr.nn0 = predict(nn, test.data)
# Test MSE
(MSE.nn.1 = RMSE(test.data$medv*sd+mean, pr.nn0*sd+mean)^2)
```

```
## [1] 42.90577
```

3. NN model with (i) one hidden layer with 3 neurons, (ii) the default loss function of ‘sse’, and (iii) the default activation function of ‘identity’.

```
set.seed(123)
nn = neuralnet(medv~., data = train.data, hidden = 3, err.fct = "sse", linear.output = T)
plot(nn, rep = 'best')
```




```
pr.nn1 = predict(nn, test.data)
# Test MSE
(MSE.nn.2 = RMSE(test.data$medv*sd+mean, pr.nn1*sd+mean)^2)
```

```
## [1] 37.01437
```

4. MLR model

```
set.seed(123)
mlr = lm(medv~., data = train.data)
summary(mlr)
```

```
##
## Call:
## lm(formula = medv ~ ., data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.40013 -0.26214 -0.06769  0.17814  2.35738
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.035423   0.023695  -1.495 0.135789
## crim        -0.042440   0.031485  -1.348 0.178506
## zn           0.058543   0.035236   1.661 0.097474 .
## indus       -0.044564   0.046408  -0.960 0.337545
## chas         0.048801   0.026021   1.875 0.061518 .
## nox         -0.181812   0.050075  -3.631 0.000323 ***
## rm           0.327836   0.032518  10.082  < 2e-16 ***
## age         -0.009319   0.042395  -0.220 0.826144
## dis         -0.259040   0.045304  -5.718 2.23e-08 ***
## tax         -0.031723   0.043656  -0.727 0.467900
## ptratio     -0.197012   0.030720  -6.413 4.38e-10 ***
## black        0.074621   0.026127   2.856 0.004533 **
## lstat       -0.324002   0.044322  -7.310 1.68e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4609 on 368 degrees of freedom
## Multiple R-squared:  0.7626, Adjusted R-squared:  0.7549
## F-statistic: 98.52 on 12 and 368 DF,  p-value: < 2.2e-16
```

```r
pr.mlr = predict(mlr, test.data)
# Test MSE
(MSE.mlr = RMSE(test.data$medv*sd+mean, pr.mlr*sd+mean)^2)
```

```
## [1] 42.90551
```

```r
# Compare MSE
print(paste(MSE.nn.1, MSE.nn.2, MSE.mlr))
```

```
## [1] "42.9057695828218 37.0143749146895 42.9055116896361"
```

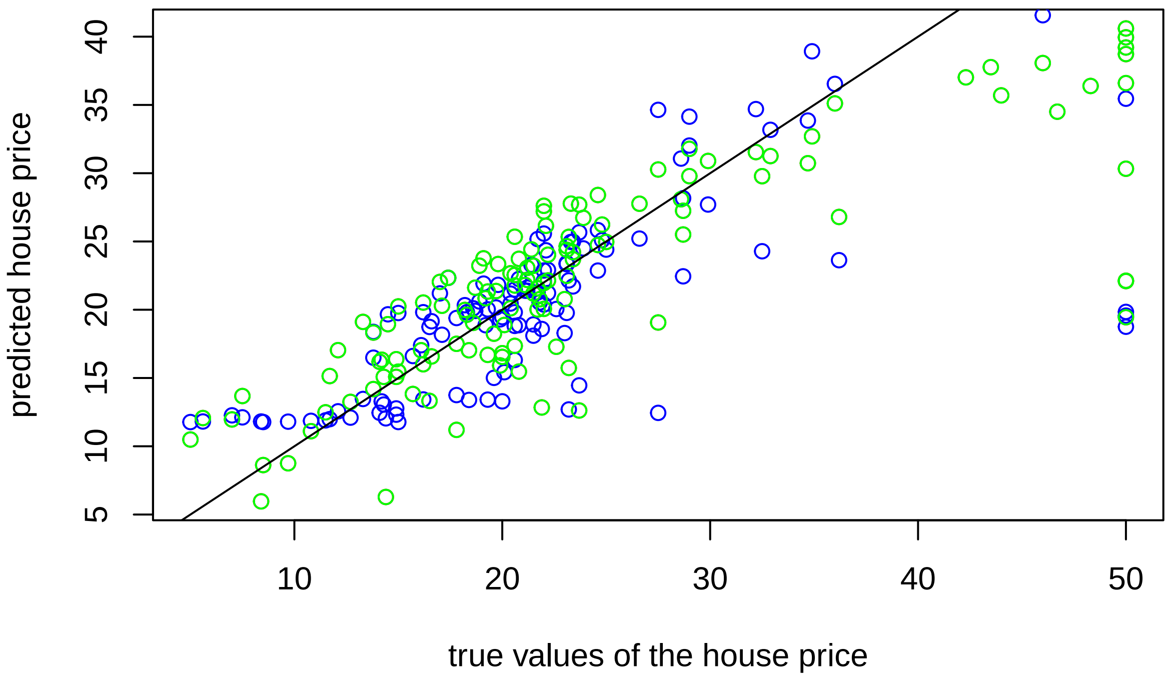
```r
# Compare with multiple linear regression
# summarize the predictions from different models
final1 <- data.frame(predictions_NN0=pr.nn0*sd+mean, predictions_NN1=pr.nn1*sd+mean,predictions_MLR =pr
knitr::kable(head(final1))
```

	predictions_NN0	predictions_NN1	predictions_MLR	actual_response
3	30.73145	33.85545	30.73168	34.7
6	25.50940	22.44689	25.50949	28.7
9	13.32310	18.73894	13.32406	16.5
11	20.24198	19.76108	20.24290	15.0

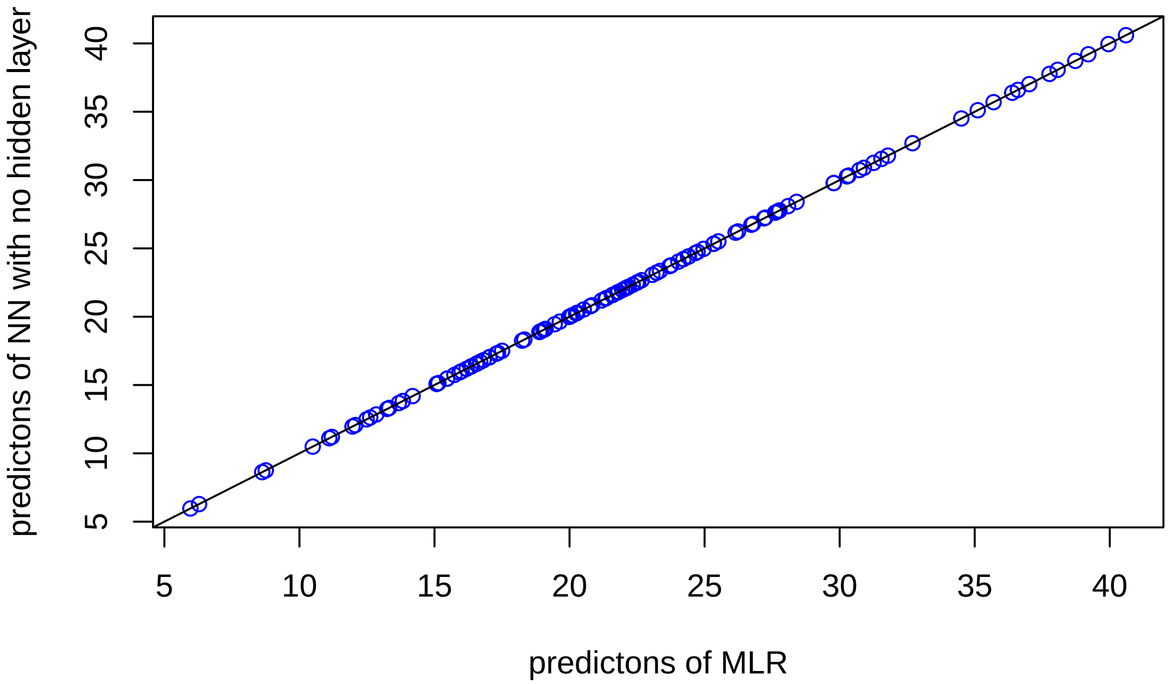
	predictions_NN0	predictions_NN1	predictions_MLR	actual_response
14	20.10770	20.45413	20.10700	20.4
15	19.98503	20.33412	19.98474	18.2

```
attach(final1)

# NN model with no hidden layer, with one hidden layer with 3 neurons and MLR vs. the true values
plot(actual_response,predictions_NN0,col="red", ylab = 'predicted house price', xlab = 'true values of t
points(actual_response,predictions_NN1,col="blue")
points(actual_response,predictions_MLR,col="green")
abline(a = 0, b = 1)
```




```
# NN model with one hidden layer with 3 neurons vs. MLR
plot(predictions_MLR,predictions_NN0,col="blue", ylab = 'predictons of NN with no hidden layer', xlab =
abline(a = 0, b = 1)
```



5. NN model with (i) one hidden layer with 3 neurons, (ii) the default loss function of "sse", and (iii) the output layer with the default activation function of 'identity', but the hidden layer with the activation function of 'relu'.

```
# devtools::install_github('rstudio/cloudml')
library(keras)
library(dplyr)
library(cloudml)
```

```
## Loading required package: tfruns
```

```
train_x = subset(train.data, select = -medv)
train_x_s = scale(train_x)
train_y = as.matrix(subset(train.data, select = medv))
test_x = subset(test.data, select = -medv)
test_x_s = scale(test_x)
test_y = as.matrix(subset(test.data, select = medv))

set.seed(123)
model <- keras_model_sequential()
```

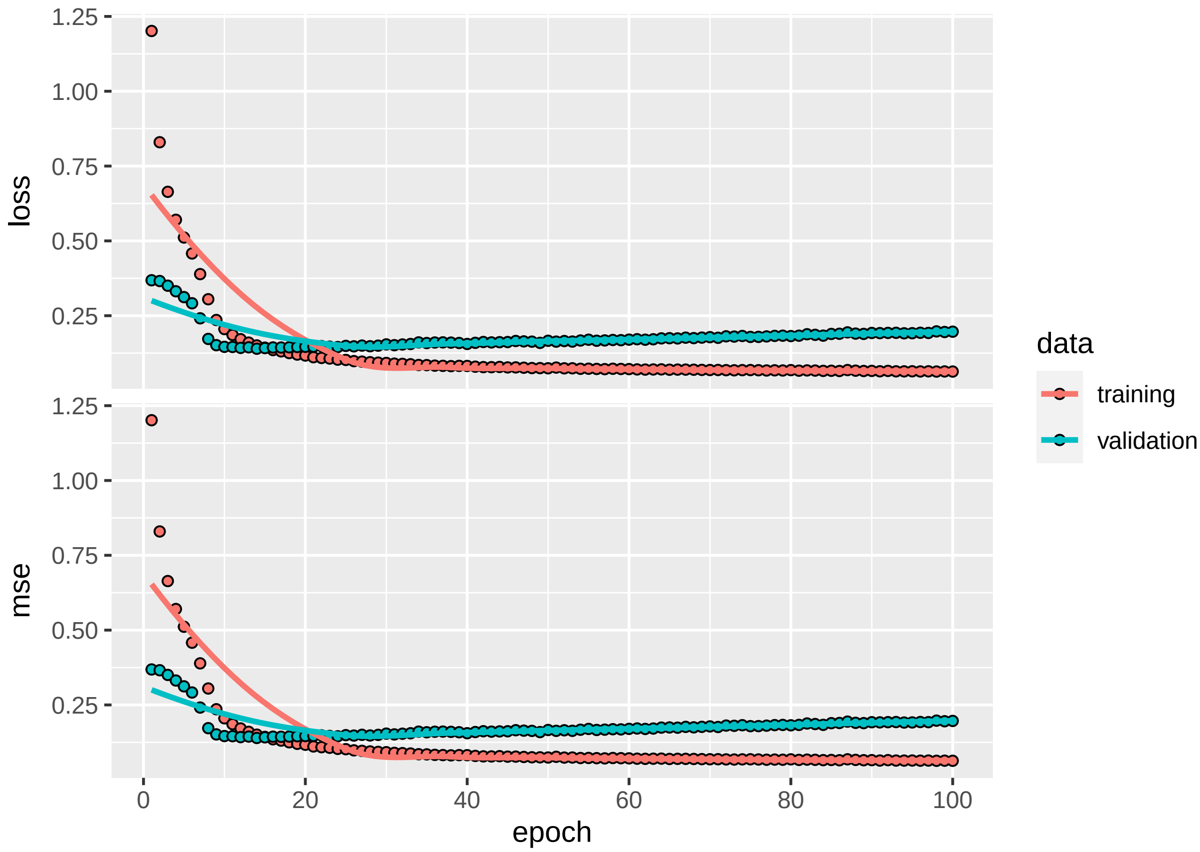
```
## Loaded Tensorflow version 2.4.4
```

```
model %>% layer_dense(units = 12, activation = 'relu', input_shape = c(12)) %>%
  layer_dense(units = 3, activation = "relu") %>%
  layer_dense(units = 1, activation = "linear")
model %>% compile(loss='mse',optimizer='adam',metrics='mse')
summary(model)
```

```
## Model: "sequential"
## ________________________________________________________________________________
## Layer (type)                        Output Shape                     Param #
## ================================================================================
## dense_2 (Dense)                     (None, 12)                       156
## ________________________________________________________________________________
## dense_1 (Dense)                     (None, 3)                        39
## ________________________________________________________________________________
## dense (Dense)                       (None, 1)                        4
## ================================================================================
## Total params: 199
## Trainable params: 199
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
history = model %>% fit(train_x_s,train_y, epochs=100,batch_size = 8,validation_split = 0.1)
plot(history)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
preds <- predict(model, test_x_s)

# test MSE
RMSE(test.data$medv*sd+mean, preds*sd+mean)^2
```

```
## [1] 16.63245
```

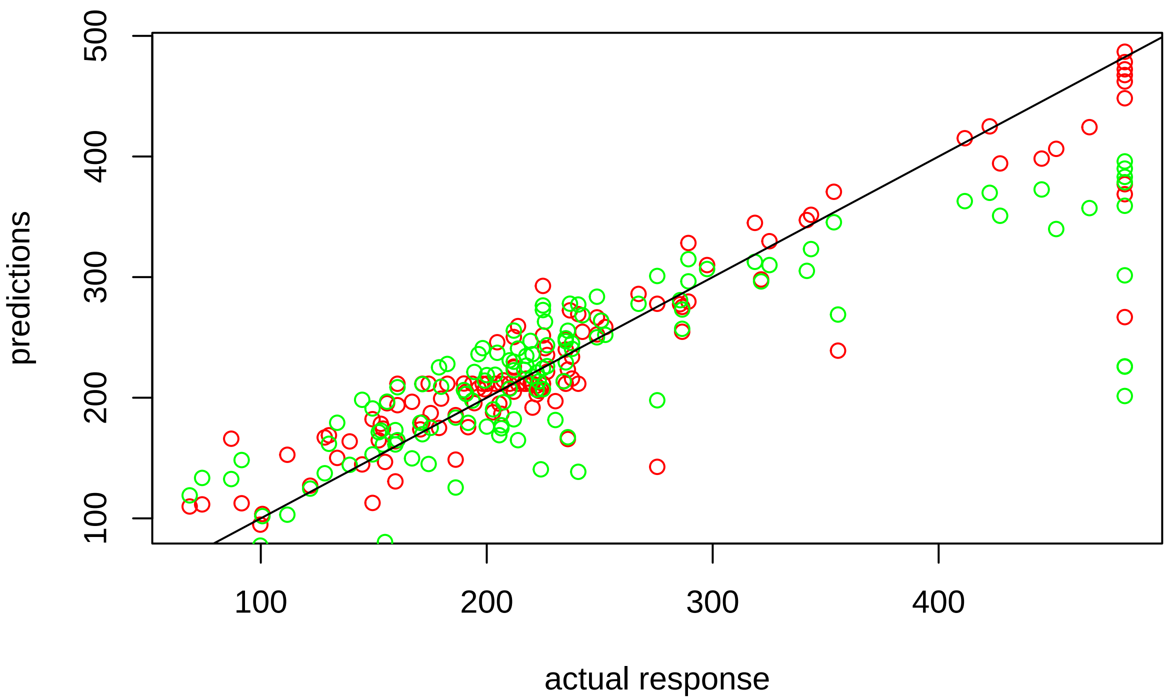
```r
# Compare with multiple linear regression
final2 <- data.frame(predictions_NN_RELU=preds*sd+mean,predictions_MLR =pr.mlr*sd+mean, actual_response=
knitr::kable(head(final2))
```

	predictions_NN_RELU	predictions_MLR	medv
3	35.31134	30.73168	34.7
6	25.24495	25.50949	28.7
9	20.55788	13.32406	16.5
11	20.55788	20.24290	15.0
14	20.16028	20.10700	20.4
15	20.57497	19.98474	18.2

```r
attach(final2)
```

```
## The following object is masked from final1:
##
##     predictions_MLR
```

```r
plot(actual_response*sd+mean,predictions_NN_RELU*sd+mean,col="red", ylab = 'predictions', xlab = 'actual
points(actual_response*sd+mean,predictions_MLR*sd+mean,col="green")
abline(a = 0, b = 1)
```



guidance of how to deal with binary string variables

The variable 'w4' in Quiz 7 is a string variable (Y/N), you can clean it in this way

```r
(w4 = c(rep('Y',5), rep('N',5)))
```

```
##  [1] "Y" "Y" "Y" "Y" "Y" "N" "N" "N" "N" "N"
```

```r
# replace 'Y' with 1 and 'N' with 0:
library(dplyr)
(w4 = ifelse(w4 == 'N', 0,1))
```

```
##  [1] 1 1 1 1 1 0 0 0 0 0
```

keras example https://www.datatechnotes.com/2019/01/regression-example-with-keras-in-r.html

python installation https://www.dataquest.io/blog/installing-python-on-mac/ https://phoenixnap.com/kb/how-to-install-python-3-windows