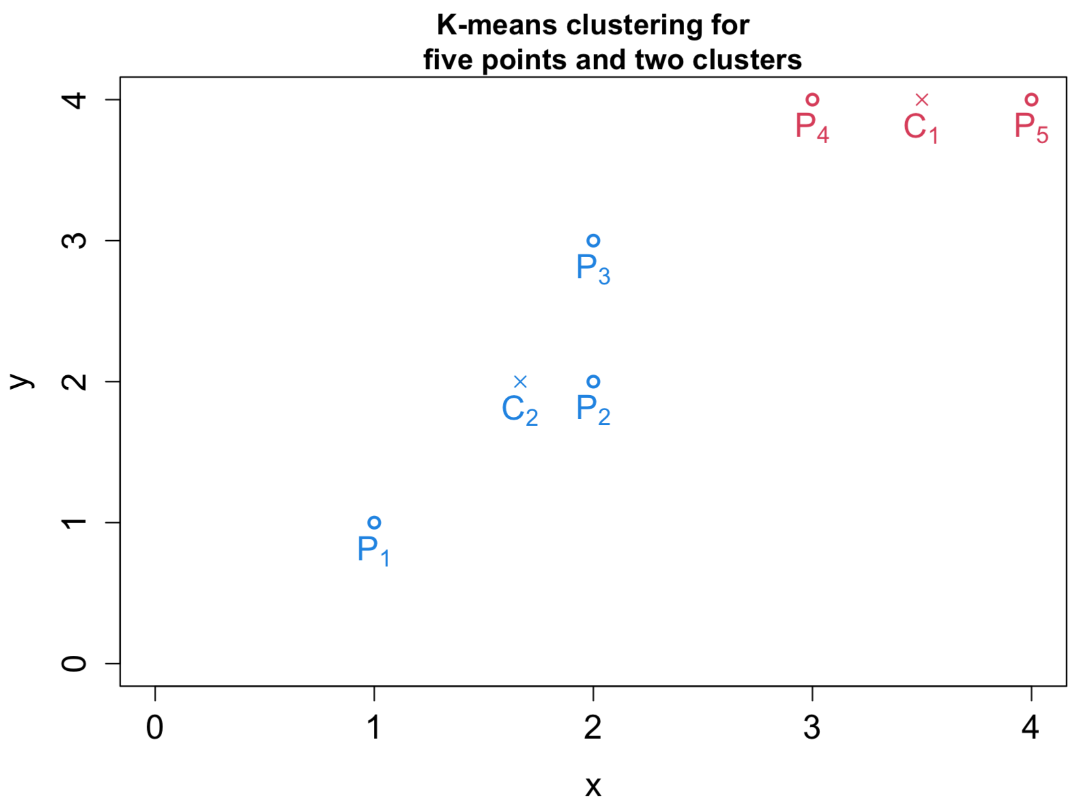
MATH524 Assignment 3: Solution by Zino Meyer ID 132611593

December 10, 2024

3.1)

```
# 3.1
# Kmeans clustering for N = 5 and K = 2
N <- 5
K <- 2
mydata <- matrix(
     c(
          1, 1,
          2, 2,
          2, 3,
          3, 4,
          4, 4
     ),
     nrow = N, byrow = TRUE
)
mydata
# [,1] [,2]
# [1,]    1    1
# [2,]    2    2
# [3,]    2    3
# [4,]    3    4
# [5,]    4    4

Kclusters <- kmeans(mydata, K)
Kclusters
# K-means clustering with 2 clusters of sizes 2, 3
#
# Cluster means:
#   [,1] [,2]
# 1 3.500000    4
# 2 1.666667    2
#
# Clustering vector:
#   [1] 2 2 2 1 1
#
```

```
# Within cluster sum of squares by cluster:
#   [1] 0.500000 2.666667
# (between_SS / total_SS =  73.6 %)

Kclusters$tot.withinss
# [1] 3.166667

cluster <- Kclusters$cluster
# [1] 2 2 2 1 1

centers <- Kclusters$centers
C1 <- centers[1, ]
C2 <- centers[2, ]

# R plot K-means
par(mar = c(4, 4, 2.5, 0.5))
plot(mydata[, 1], mydata[, 2],
     lwd = 2,
     xlim = c(0, 4), ylim = c(0, 4),
     xlab = "x", ylab = "y",
     col = cluster * 2,
     main = "K-means clustering for
     five points and two clusters",
     cex.lab = 1.4, cex.axis = 1.4
)
points(centers[, 1], centers[, 2],
     col = c(2, 4), pch = 4
)
for (i in K:1) {
     text(centers[i, 1], centers[i, 2] - 0.2,
          bquote(C[.(i)]),
          cex = 1.4, col = i * 2)
}
for (i in 1:N) {
     text(mydata[i, 1], mydata[i, 2] - 0.2,
          bquote(P[.(i)]),
          cex = 1.4, col = cluster[i] * 2)
}

dev.off()
```



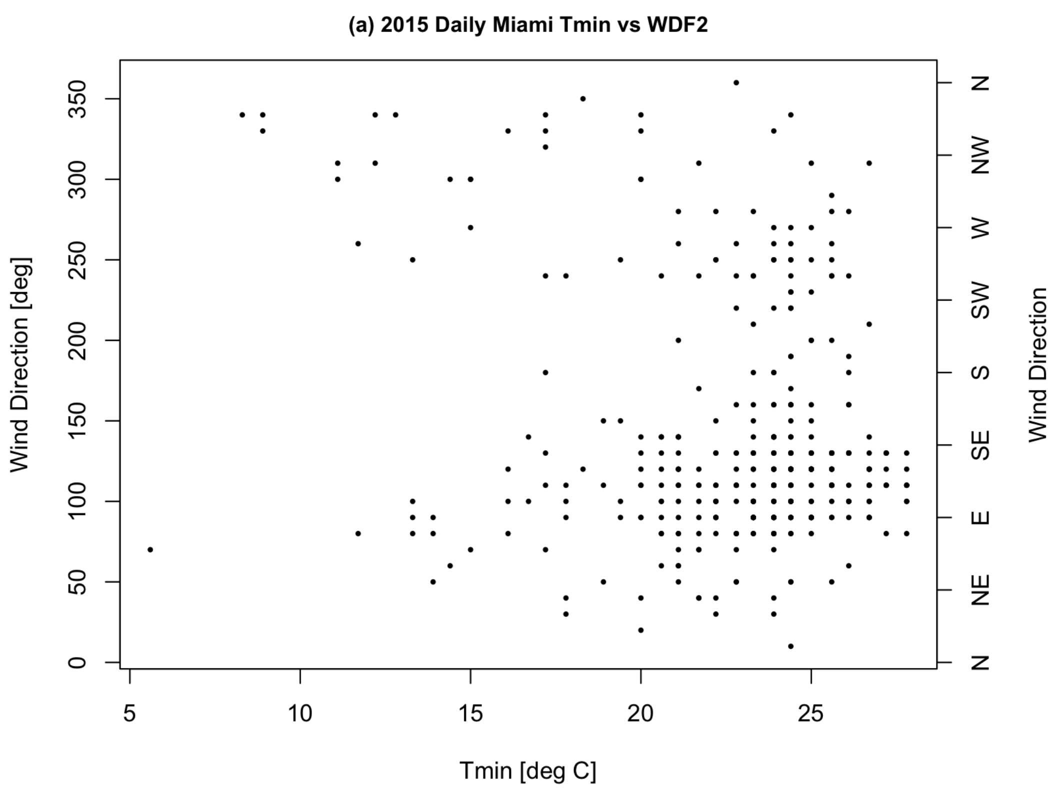
As seen in the code, the tWCSS (tot.withinss) is 3.166667.

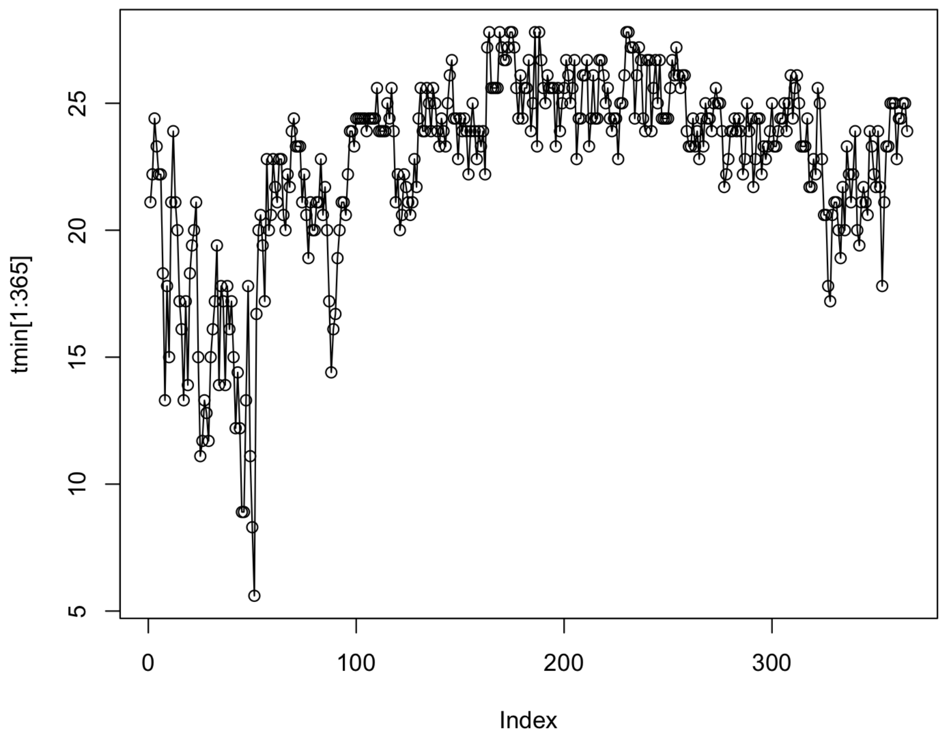
3.3)

```
# 3.3
# Kmeans clustering for MiamiIntl data
data = read.csv("data/MiamiIntlAirport2001_2020.csv",
               header=TRUE)
dim(data)
#[1] 7305   29

data_2015 <- data[5114: 5478, ]
dim(data_2015)
# [1] 365  29

tmin = data_2015[,'TMIN']
wdf2 = data_2015[,'WDF2']

par(mar=c(4.5, 4.5, 2, 4.5))
plot(tmin[1:365], wdf2[1:365],
     pch =16, cex = 0.5,
     xlab = 'Tmin [deg C]',
```

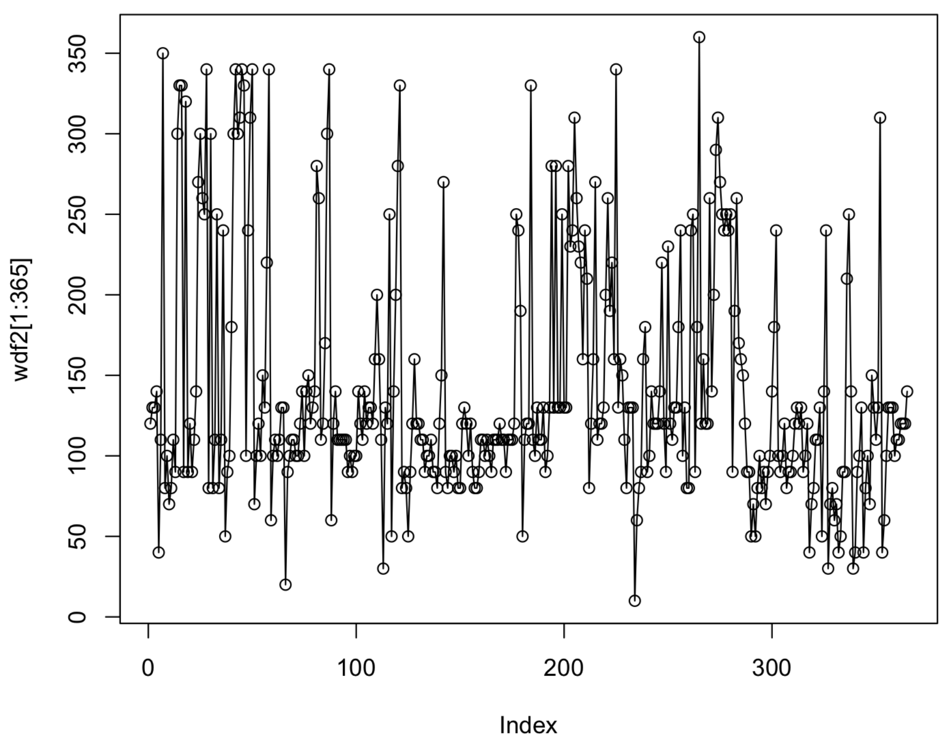
```
     ylab = 'Wind Direction [deg]',
     type = 'p')
title('(a) 2015 Daily Miami Tmin vs WDF2', cex.main = 0.9, line = 1)
axis(4, at = c(0, 45, 90, 135, 180, 225, 270, 315, 360),
     lab = c('N', 'NE', 'E', 'SE', 'S', 'SW',  'W', 'NW', 'N'))
mtext('Wind Direction', side = 4, line =3)

dev.off()

# Plot time series of temperature & wind direction
plot(tmin[1:365], type="o")
plot(wdf2[1:365], type="o")

#K-means clustering
K = 2 # assuming 2 clusters
mydata = cbind(tmin[1:365], wdf2[1:365])
fit = kmeans(mydata, K)

# Output total sum of squares tSS
fit$totss
# [1] 2044891

#Output the coordinates of the cluster centers
fit$centers
#   [,1]     [,2]
# 1 21.08481 268.8608
# 2 23.07797 106.8531

# Output total within-cluster sum of squares tWCSS
fit$tot.withinss
# [1] 419952.8

#Visualize the clusters by kmeans.ani()
mycluster <- data.frame(mydata, fit$cluster)
names(mycluster)<-c('Tmin [deg C]',
                    'Wind Direction [deg]',
                    'Cluster')

library(animation)
kmeans_animation <- function() {
  par(mar = c(4.5, 4.5, 2, 4.5))
  kmeans.ani(mycluster, centers = K, pch=1:K, col=1:K, hints = '')
  title(main = "(b) K-means Clusters for Daily Tmin vs WDF2 in 2015",
        cex.main = 0.8)
  axis(4, at = c(0, 45, 90, 135, 180, 225, 270, 315, 360),
```

```
      lab = c('N', 'NE', 'E', 'SE', 'S', 'SW',  'W', 'NW', 'N'))
  mtext('Wind Direction', side = 4, line = 3)
}

# Run this to see the animation
kmeans_animation()

# Run this to save the animation
# saveGIF(kmeans_animation(),
#         movie.name = "kmeans_anim_miami_temp_wind_2015.gif",
#         interval = 0.1)
```





25
20
15
10
5
tmin[1:365]
0
100
200
300
Index

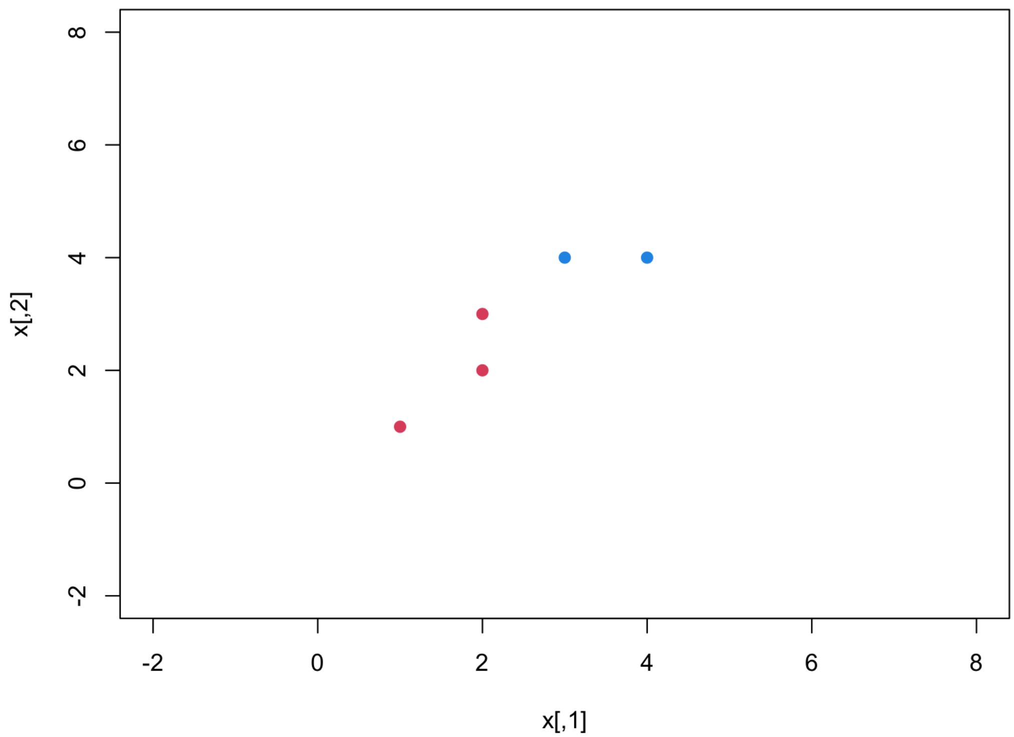


350
300
250
200
150
100
50
0
wdf2[1:365]
0
100
200
300
Index

3.8)

```
# 3.8
# SVM classification for five points
N <- 5
x <- matrix(
  c(
    1, 1,
    2, 2,
    2, 3,
    3, 4,
    4, 4
  ),
  nrow = N, byrow = TRUE
)
y = c(1, 1, 1, 2, 2) # two categories 1 and 2

# Plot points
plot(x, col = y * 2, pch = 19,
     xlim = c(-2, 8), ylim = c(-2, 8))

# Train SVM
```

```
library(e1071)
dat = data.frame(x, y = as.factor(y))
svm5P = svm(y ~ ., data = dat,
            kernel = "linear", cost = 10,
            scale = FALSE,
            type = 'C-classification')

# Find hyperplane, normal vector, and SV (wx + b = 0)
w <- t(svm5P$coefs) %*% svm5P$SV
w
# [1,] -1 -1

b <- svm5P$rho
b
# [1] -6

# Maximum margin of separation
Dm <- 2/norm(w, type ='2')
Dm
# [1] 1.414214

# Support vectors
SV <- svm5P$SV
SV
# X1 X2
# 3  2  3
# 4  3  4
```



3.9)

```
# 3.9
# Predict the category of two new points
x_new <- matrix(c(1.5, 1, 3, 3),
     ncol = 2, byrow = TRUE
)

result <- data.frame(x_new, predict(svm5P, x_new))
result
```

3.13)

```
# 3.13
data(iris)
dim(iris)
# [1] 150   5

# select rows 1-40, 51-90, and 101-140 as training data
train_data <- rbind(iris[1:40, ], iris[51:90, ], iris[101:140, ])
dim(train_data)
```

```
# [1] 120   5

library(randomForest)
set.seed(42)

# select the remaining data as test data
test_data <- rbind(iris[41:50, ], iris[91:100, ], iris[141:150, ])
dim(test_data)
# [1] 30  5

# train the RF model
classifyRF = randomForest(x = train_data[, 1:4],
                          y = train_data[, 5], ntree = 800)
classifyRF
# Type of random forest: classification
# Number of trees: 800
# No. of variables tried at each split: 2
#
# OOB estimate of  error rate: 5%
# Confusion matrix:
#   setosa versicolor virginica class.error
# setosa         40          0         0       0.000
# versicolor      0         37         3       0.075
# virginica       0          3        37       0.075

# RF prediction for the test data
predict(classifyRF, test_data[,1:4])
# 41         42         43         44         45
# setosa     setosa     setosa     setosa     setosa
# 46         47         48         49         50
# setosa     setosa     setosa     setosa     setosa

# 91         92         93         94         95
# versicolor versicolor versicolor versicolor versicolor
# 96         97         98         99         100
# versicolor versicolor versicolor versicolor versicolor

# 141        142        143        144        145
# virginica  virginica  virginica  virginica  virginica
# 146        147        148        149        150
# virginica  virginica  virginica  virginica  virginica

# Levels: setosa versicolor virginica
```

3.14)

```
# 3.14
set.seed(42)
# select random indices of 20% of data from each species as training data
train_ids <- c(
     sample(1:50, 10),
     sample(51:100, 10),
     sample(101:150, 10)
)
train_data <- iris[train_ids, ]
dim(train_data)
# [1] 30  5

# train the RF model
classifyRF <- randomForest(
     x = train_data[, 1:4],
     y = train_data[, 5], ntree = 800
)
classifyRF
# Type of random forest: classification
# Number of trees: 800
# No. of variables tried at each split: 2
#
# OOB estimate of  error rate: 10%
# Confusion matrix:
#   setosa versicolor virginica class.error
# setosa         10          0         0         0.0
# versicolor      0          9         1         0.1
# virginica       0          2         8         0.2

# select random indices of 10% of data from each species as test data
test_ids <- sample(1:150, 15)
test_data <- iris[test_ids, ]
dim(test_data)
# [1] 15  5

# RF prediction for the test data
predict(classifyRF, test_data[, 1:4])
# 4       15      148         24      2
# setosa  setosa  virginica   setosa  setosa

# 118         23          51               7
# virginica   setosa      versicolor     setosa
```

```
# 61          132        116         30       145        134
# versicolor  virginica  virginica   setosa   virginica  virginica
```

The error rate of the RF model is higher than in the previous exercise. The classification error for versicolor is 0.1, previously was 0.075, and for virginica is 0.2, previously also 0.075.

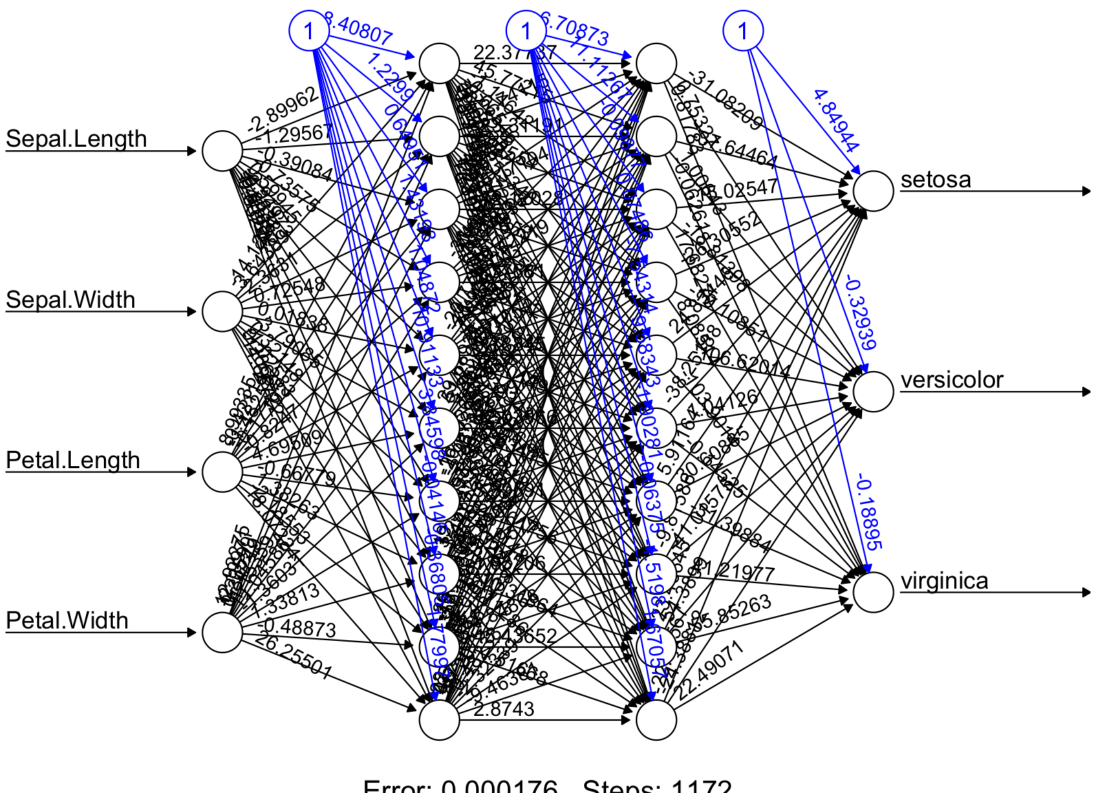
But the difference is not that dramatic. Is is still a model that can differentiate between three species. Looking at the prediction, it successfully is able to classify all test data samples.

This insight means that with random forest classification, the dataset does not have to be large if the data is well separated, as in the case of the leave length of iris flowers. It is fairly easy for the decision trees to make a decision based on the length.

3.19)

```
# 3.19
# Attach True or False columns to iris data
iris$setosa = iris$Species == "setosa"
iris$virginica = iris$Species == "virginica"
iris$versicolor = iris$Species == "versicolor"

# select rows 1-40, 51-90, and 101-140 as training data
train_data <- rbind(iris[1:40, ], iris[51:90, ], iris[101:140, ])
dim(train_data)
# [1] 120   8


# select the remaining data as test data
test_data <- rbind(iris[41:50, ], iris[91:100, ], iris[141:150, ])
dim(test_data)
# [1] 30  8

library(neuralnet)

# use the length, width, True and False data for training
iris.nn = neuralnet(setosa + versicolor + virginica ~
                      Sepal.Length + Sepal.Width +
                      Petal.Length + Petal.Width,
                    data = train_data, hidden=c(10, 10),
                    rep = 5, err.fct = "ce",
                    linear.output = F, lifesign = "minimal",
                    stepmax = 1000000, threshold = 0.001)

plot(iris.nn, rep="best") #plot the neural network
```

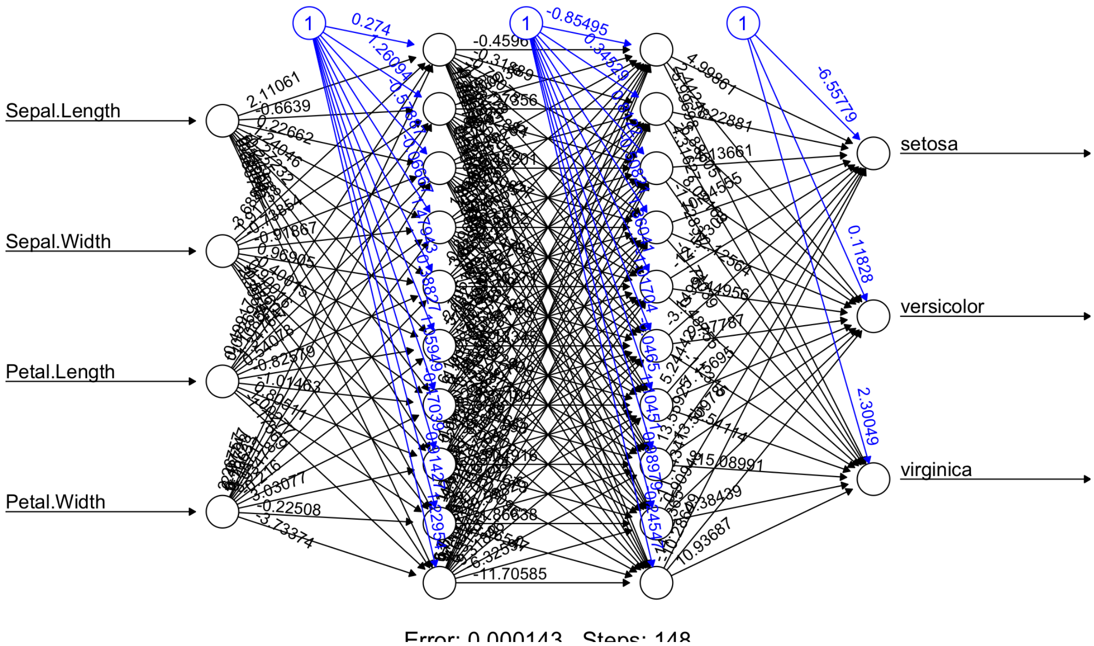
```
# Prediction for the rest data
prediction = neuralnet::compute(iris.nn, test_data[,1:4])

#print the first 3 rows
prediction$net.result[1:3,]
#    [,1]         [,2]         [,3]
# 41    1 3.630561e-17 4.502644e-82
# 42    1 3.373191e-18 4.633479e-81
# 43    1 1.687628e-16 1.028237e-82

# Find which column is for the max of each row
pred.idx <- apply(prediction$net.result, 1, which.max)

# The prediction result: Assign 1 for setosa, 2 for versicolor, 3 for virginica
predicted <- c('setosa', 'versicolor', 'virginica')[pred.idx]
predicted
# [1] "setosa"     "setosa"     "setosa"     "setosa"     "setosa"     "setosa"
# [8] "setosa"     "setosa"     "setosa"     "versicolor" "versicolor" "versicol
# [15] "versicolor" "versicolor" "versicolor" "versicolor" "versicolor" "versico
# [22] "virginica"  "virginica"  "virginica"  "virginica"  "virginica"  "virgini
# [29] "virginica"  "virginica"

# Create confusion matrix: table(prediction,observation)
table(predicted, test_data$Species)
# predicted    setosa versicolor virginica
# setosa         10          0         0
# versicolor      0         10         0
# virginica       0          0        10
```



3.20)

```
# 3.20
set.seed(42)
# Attach True or False columns to iris data
iris$setosa = iris$Species == "setosa"
iris$virginica = iris$Species == "virginica"
iris$versicolor = iris$Species == "versicolor"
# select random indices of 20% of data from each species as training data
train_ids <- c(
  sample(1:50, 10),
  sample(51:100, 10),
  sample(101:150, 10)
)
train_data <- iris[train_ids, ]
dim(train_data)
# [1] 30  8

# select random indices of 10% of data from each species as test data
test_ids <- sample(1:150, 15)
test_data <- iris[test_ids, ]
dim(test_data)
```

```
# [1] 15  8

# Train NN
iris.nn = neuralnet(setosa + versicolor + virginica ~
                      Sepal.Length + Sepal.Width +
                      Petal.Length + Petal.Width,
                    data = train_data, hidden=c(10, 10),
                    rep = 5, err.fct = "ce",
                    linear.output = F, lifesign = "minimal",
                    stepmax = 1000000, threshold = 0.001)

# plot the neural network
par(mar = c(8,8,8,8))
plot(iris.nn, rep="best")

# Prediction for the rest data
prediction = neuralnet::compute(iris.nn, test_data[,1:4])

#print the first 3 rows
prediction$net.result[1:3,]
#    [,1]         [,2]         [,3]
# 42    1 1.817944e-06 3.518222e-15
# 24    1 1.614523e-06 3.614067e-15
# 43    1 1.487381e-06 3.698661e-15

# Find which column is for the max of each row
pred.idx <- apply(prediction$net.result, 1, which.max)

# The prediction result: Assign 1 for setosa, 2 for versicolor, 3 for virginica
predicted <- c('setosa', 'versicolor', 'virginica')[pred.idx]
predicted
# [1] "setosa"     "setosa"     "setosa"     "virginica"  "versicolor" "virginic
# [7] "setosa"     "versicolor" "virginica"  "versicolor" "setosa"     "versicol
# [13] "virginica"  "setosa"     "virginica"

# Create confusion matrix: table(prediction,observation)
table(predicted, test_data$Species)
# predicted    setosa versicolor virginica
# setosa          6          0         0
# versicolor      0          4         0
# virginica       0          0         5
```

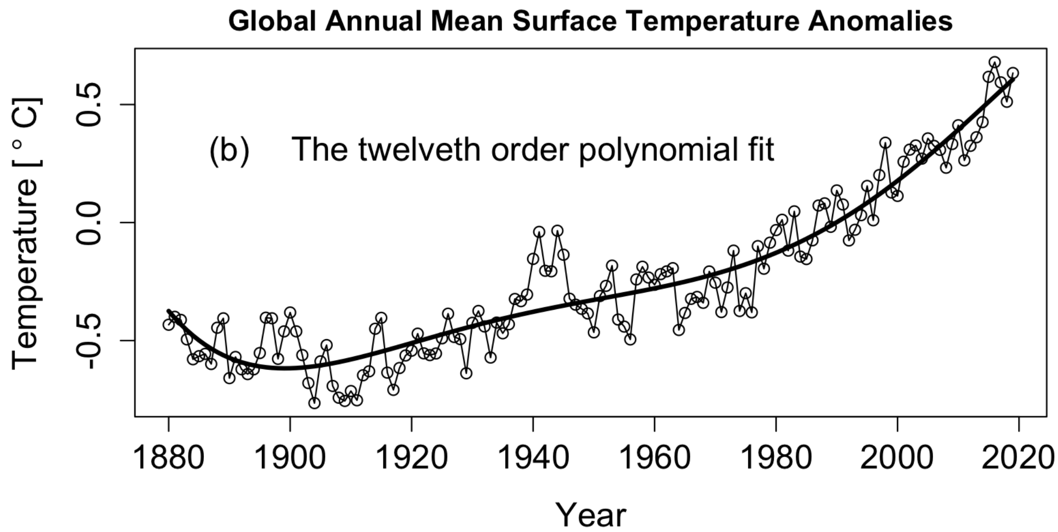


As we can see, the neural net shows even better results as in random forest: although the dataset was smaller as in the previous exercise, the results are still very clear. The neural net is also able to properly predict species based on sepal & petal width, also with a very small training data set. There is no misclassification although the training set is just 20% of the original size.

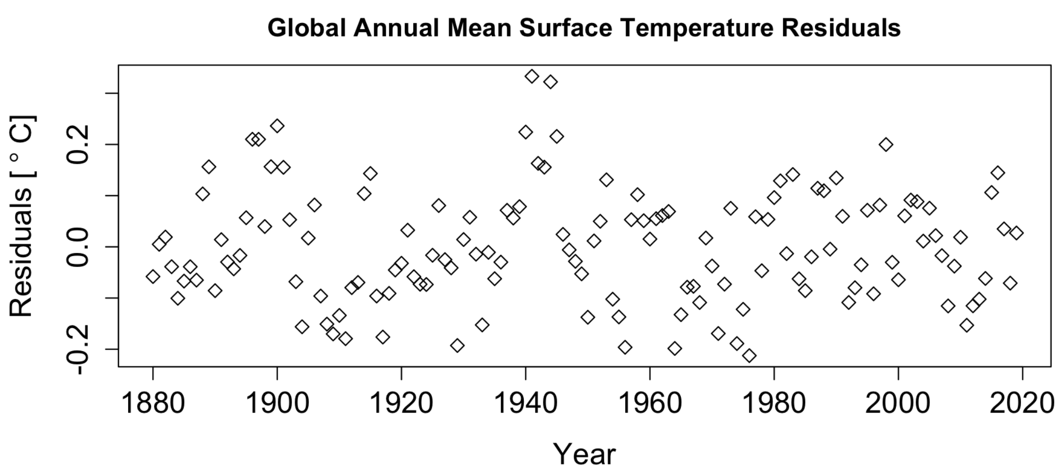
4.11)

```
# 4.11
# (a) Use multiple linear regression to compute the 12th order polynomial
# fitting of the NOAAGlobalTemp’s global average annual mean data from 1880
# to 2018:
#   T= b0 + b1t+ b2t2 +···+ b12t12 + ϵ.

data <- read.csv("data/NOAAGlobalTempAnn2019.csv", header = FALSE)

x = data[,1]
y = data[,2]

# Polynomial fitting by multiple linear  regression
x1 <- x
x2 <- x1^2
x3 <- x1^3
x4 <- x1^4
x5 <- x1^5
x6 <- x1^6
x7 <- x1^7
```

```
x8 <- x1^8
x9 <- x1^9
x10 <- x1^10
x11 <- x1^11
x12 <- x1^12

dat12 <- data.frame(cbind(x1, x2, x3, x4, x5, x6,
                          x7, x8, x9, x10, x11, x12, y))
reg12 <- lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 +
              x7 + x8 + x9 + x10 + x11 + x12, data = dat12)
reg12
# Coefficients:
# (Intercept)           x1           x2
# 6.021e+06   -1.404e+04    1.253e+01

# x3           x4           x5
# -5.108e-03    8.136e-07           NA

# x6           x7           x8
# NA           NA   -7.827e-22

# x9          x10          x11
# NA           NA           NA

# x12
# NA

# (b) Plot the data and the fitted polynomial function on the same figure.
par(mar = c(5, 5, 2, 1))
plot(x, y,
     type = "o",
     cex.lab = 1.4, cex.axis = 1.4,
     xlab = "Year",
     ylab = bquote("Temperature [" ~ degree ~ "C]"),
     main = "Global Annual Mean Surface Temperature Anomalies",
     cex.lab = 1.4, cex.axis = 1.4
)
lines(x, predict(reg12), col = "black", lwd = 3)
text(1890, 0.3, "(b)", cex = 1.4)
text(1940, 0.3,
     "The twelveth order polynomial fit",
     col = "black", cex = 1.4
)
```



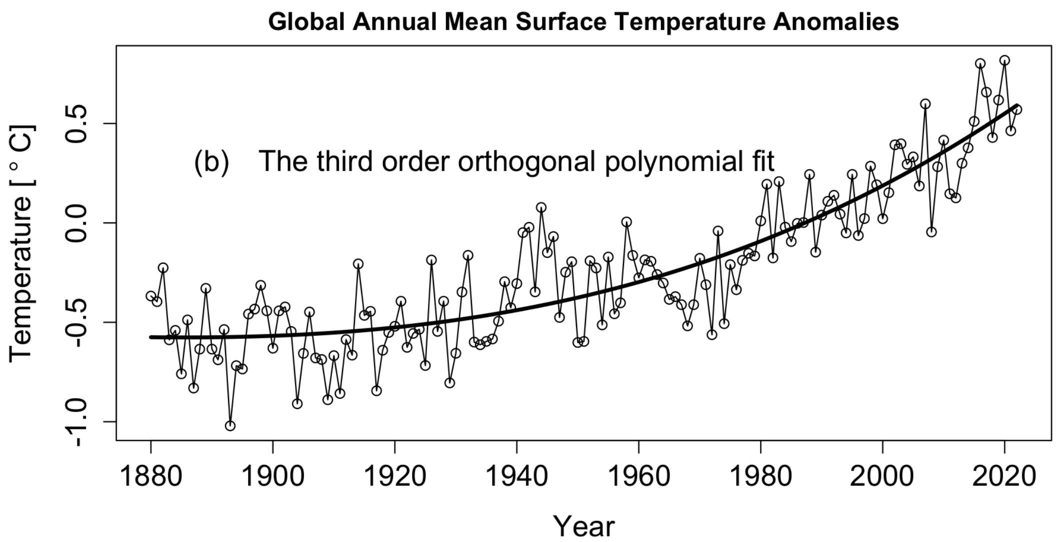

```
# (c) Produce a scatter plot of the residuals of the fitting against time.
par(mar = c(4.5, 4.5, 0, 0.7))
plot(x1, reg12$residuals,
     pch = 5, cex.lab = 1.4, cex.axis = 1.4,
     xlab = "Year", ylab = bquote("Residuals [" ~ degree ~ "C]")
)
text(1880, 0.32, "(b)", cex = 1.4)
```



4.13)

```
# 4.13
```

```r
# (a) Fit the global average January monthly mean temperature anomaly
# data from 1880 to 2018 in the NOAAGlobalTemp dataset to a third order
# orthogonal polynomial.

data <- read.table("data/NOAAGlobalTmonthly.txt", header = FALSE)

# filter for january data
data <- data[data[, 2] == 1, ]

x <- data[, 1]
y <- data[, 3]

# Polynomial fitting by multiple linear  regression
x1 <- x
x2 <- x1^2
x3 <- x1^3

dat3 <- data.frame(cbind(x1, x2, x3, y))
reg3 <- lm(y ~ x1 + x2 + x3, data = dat3)
reg3
# Coefficients:
# (Intercept)           x1           x2           x3
#   6.500e+00   -1.049e-08    5.377e-12   -9.184e-16

# (b) Plot the data and the fitted polynomial function on the same figure.
par(mar = c(5, 5, 2, 1))
plot(x, y,
     type = "o",
     cex.lab = 1.4, cex.axis = 1.4,
     xlab = "Year",
     ylab = bquote("Temperature [" ~ degree ~ "C]"),
     main = "Global Annual Mean Surface Temperature Anomalies",
     cex.lab = 1.4, cex.axis = 1.4
)
lines(x, predict(reg3), col = "black", lwd = 3)
text(1890, 0.3, "(b)", cex = 1.4)
text(1940, 0.3,
     "The third order orthogonal polynomial fit",
     col = "black", cex = 1.4
)
```




```
# (c) Produce a scatter plot of the residuals of the fitting against
# time in another figure.
par(mar = c(5, 5, 3, 1))
plot(x1, reg3$residuals,
     pch = 5, cex.lab = 1.4, cex.axis = 1.4,
     xlab = "Year", ylab = bquote("Residuals [" ~ degree ~ "C]"),
     main = "Global Annual Mean Surface Temperature Residuals",
)
```