Chapter 15 지도학습

Sangkon Han(sangkon@pusan.ac.kr)

2023-06-29

Contents

단순 선형 회귀분석 수행

단계 1: 데이터 가져오기

```
product <- read.csv("data/product.csv", header = TRUE, fileEncoding = "euc-kr")
str(product)
```

```
## 'data.frame':    264 obs. of  3 variables:
##  $ 제품_친밀도: int  3 3 4 2 2 3 4 2 3 4 ...
##  $ 제품_적절성: int  4 3 4 2 2 3 4 2 2 2 ...
##  $ 제품_만족도: int  3 2 4 2 2 3 4 2 3 3 ...
```

단계 2: 독립변수와 종속벼수 생성

```
y = product$제품_만족도
x = product$제품_적절성
df <- data.frame(x, y)
```

단계 3: 단순 선형회귀 모델 생성

```
result.lm <- lm(formula = y ~ x, data = df)
```

단계 4: 회귀분석의 절편과 기울기

```
result.lm
```

```
##
## Call:
## lm(formula = y ~ x, data = df)
##
## Coefficients:
## (Intercept)            x
##      0.7789       0.7393
```

단계 5: 모델의 적합값과 잔차 보기

```
names(result.lm)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

단계 5-1: 적합값 보기

```
fitted.values(result.lm)[1:2]
```

```
##        1        2
## 3.735963 2.996687
```

단계 5-2: 관측값 보기

```
head(df, 1)
```

```
##   x y
## 1 4 3
```

단계 5-3: 회귀방정식을 적용하여 모델의 적합값 계산

```
Y = 0.7789 + 0.7393 * 4
Y
```

```
## [1] 3.7361
```

단계 5-4: 잔차(오차) 계산

```
3 - 3.735963
```

```
## [1] -0.735963
```

단계 5-5: 모델의 잔차 보기

```
residuals(result.lm)[1:2]
```

```
##          1          2
## -0.7359630 -0.9966869
```

단계 5-6: 모델의 잔차와 회귀방정식에 의한 적합값으로부터 관측값 계산

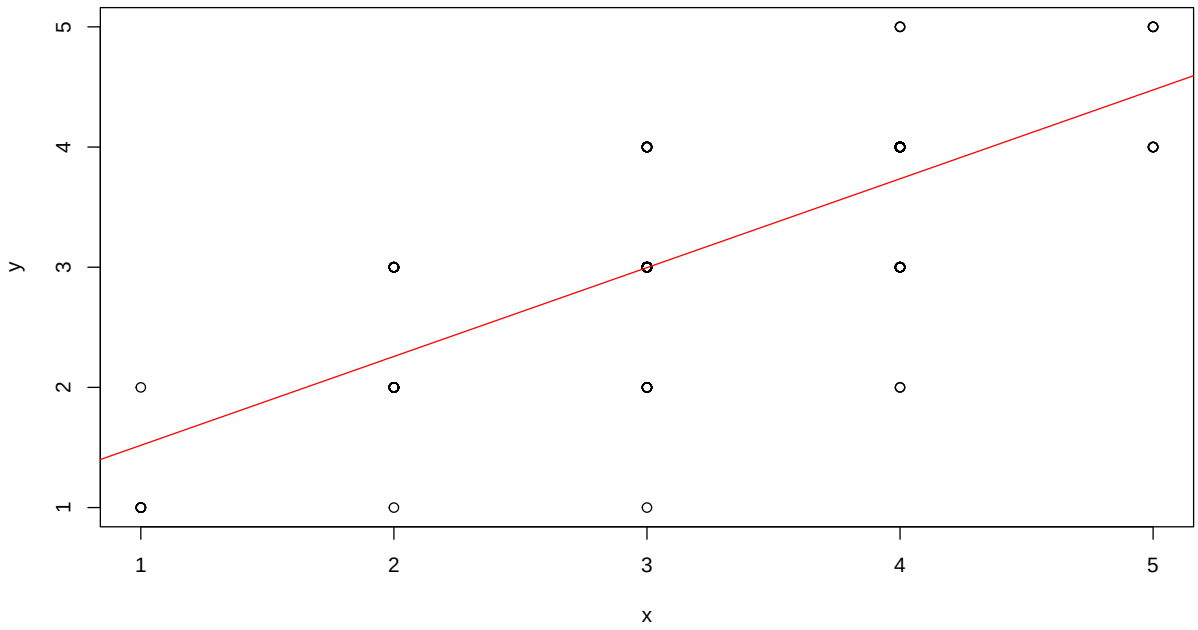
```
-0.7359630 + 3.735963
```

```
## [1] 3
```

선형 회귀분석 모델 시각화

x, y 산점도 그리기

```
plot(formula = y ~ x, data = product)
result.lm <- lm(formula = y ~ x, data = product) # 선형 회귀모델 생성
abline(result.lm, col = "red") # 회귀선
```



선형 회귀분석 결과보기

```
summary(result.lm)
```

```
##
## Call:
## lm(formula = y ~ x, data = product)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.99669 -0.25741  0.00331  0.26404  1.26404
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.77886    0.12416   6.273 1.45e-09 ***
```

```
## x            0.73928    0.03823  19.340  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5329 on 262 degrees of freedom
## Multiple R-squared:  0.5881, Adjusted R-squared:  0.5865
## F-statistic:   374 on 1 and 262 DF,  p-value: < 2.2e-16
```

다중 회귀분석

단계 1: 변수 모델링

```
y = product$제품_만족도
x1 = product$제품_친밀도
x2 = product$제품_적절성
df <- data.frame(x1, x2, y)
```

단계 2: 다중 회귀분석

```
result.lm <- lm(formula = y ~x1 + x2, data = df)
result.lm
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = df)
##
## Coefficients:
## (Intercept)           x1           x2
##     0.66731      0.09593      0.68522
```

다중 공선성 문제 확인

단계 1: 라이브러리 호출

```
#install.packages("car")
library(car)
```

```
## Loading required package: carData
```

단계 2: 분산팽창요인(VIF)

```
vif(result.lm)
```

```
##       x1       x2
## 1.331929 1.331929
```

다중 회귀분석 결과보기

```
summary(result.lm)
```

```
##
## Call:
```

```
## lm(formula = y ~ x1 + x2, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.01076 -0.22961 -0.01076  0.20809  1.20809
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.66731    0.13094   5.096 6.65e-07 ***
## x1           0.09593    0.03871   2.478   0.0138 *
## x2           0.68522    0.04369  15.684  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5278 on 261 degrees of freedom
## Multiple R-squared:  0.5975, Adjusted R-squared:  0.5945
## F-statistic: 193.8 on 2 and 261 DF,  p-value: < 2.2e-16
```

다중 공선성 문제 확인

단계 1: 패키지 설치 및 데이터 로딩

```
library(car)
data(iris)
```

단계 2: iris 데이터 셋으로 다중 회귀분석

```
model <- lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width, data = iris)
vif(model)
```

```
##  Sepal.Width Petal.Length  Petal.Width
##     1.270815    15.097572    14.234335
```

```
sqrt(vif(model)) > 2
```

```
##  Sepal.Width Petal.Length  Petal.Width
##        FALSE         TRUE         TRUE
```

단계 3: iris 변수 간의 상관계수 구하기

```
cor(iris[ , -5])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
## Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
## Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
## Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

데이터 셋 생성과 회귀모델 생성

단계 1: 학습데이터와 검저엔이터 표본 추출

```
x <-sample(1:nrow(iris), 0.7 * nrow(iris))
train <- iris[x, ]
test <- iris[-x, ]
```

단계 2: 변수 제거 및 다중 회귀분석

```
model <- lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length, data = train)
model
```

```
##
## Call:
## lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length, data = train)
##
## Coefficients:
##  (Intercept)   Sepal.Width  Petal.Length
##       2.2611        0.5968        0.4669
```

```
summary(model)
```

```
##
## Call:
## lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.95414 -0.23930  0.02063  0.22775  0.66563
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.26111    0.28651   7.892  3.5e-12 ***
## Sepal.Width   0.59680    0.07936   7.520  2.2e-11 ***
## Petal.Length  0.46690    0.02098  22.259  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3423 on 102 degrees of freedom
## Multiple R-squared:  0.8305, Adjusted R-squared:  0.8272
## F-statistic: 249.8 on 2 and 102 DF,  p-value: < 2.2e-16
```

회귀방정식 도출

단계 1: 회귀방정식을 위한 절편과 기울기 보기

```
model
```

```
##
## Call:
## lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length, data = train)
##
## Coefficients:
```

```
##  (Intercept)   Sepal.Width  Petal.Length
##       2.2611        0.5968        0.4669
```

단계 2: 회귀방정식 도출

```
head(train, 1)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 122          5.6         2.8          4.9           2 virginica
```

다중 회귀방정식 적용

```
Y = 2.3826 +  0.5684 * 2.9 + 0.4576 * 4.6
Y
```

```
## [1] 6.13592
```

```
6.6 - Y
```

```
## [1] 0.46408
```

검정데이터의 독립변수를 이용한 예측치 생성

```
pred <- predict(model, test)
pred
```

```
##        2        3        6       10       17       23       24       25
## 4.705169 4.777840 5.382360 4.811539 5.195601 4.876492 5.024278 5.177338
##       27       28       29       30       31       46       48       55
## 5.037269 5.050260 4.943890 4.917909 4.858228 4.705169 4.824530 6.079875
##       59       62       65       66       74       75       80       84
## 6.139555 6.012477 5.672659 6.165536 6.126564 5.999486 5.446929 6.253642
##       86       92       93       97      100      109      114      115
## 6.391266 6.199235 5.680377 5.952797 5.846427 6.461109 6.087592 6.313322
##      117      120      123      132      133      134      136      138
## 6.619441 5.908552 7.060355 7.517089 6.546770 6.313322 6.899578 6.679121
##      141      142      144      149      150
## 6.725811 6.492363 6.925560 6.811472 6.432683
```

상관계수를 이용한 회귀모델 평가

```
cor(pred, test$Sepal.Length)
```

```
## [1] 0.9285687
```

회귀분석의 기본 가정 충족으로 회귀분석 수행

단계 1: 회귀모델 생성

단계 1-1: 벼수 모델링

```
formula = Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width
```

단계 1-2: 회귀모델 생성

```
model <- lm(formula = formula, data = iris)
model
```

```
##
## Call:
## lm(formula = formula, data = iris)
##
## Coefficients:
##  (Intercept)   Sepal.Width  Petal.Length   Petal.Width
##       1.8560        0.6508        0.7091       -0.5565
```

단계 2: 잔차(오차) 부석

단계 2-1: 독립성 검정 - 더빈 왓슨 값으로 확인

```
# install.packages("lmtest")
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

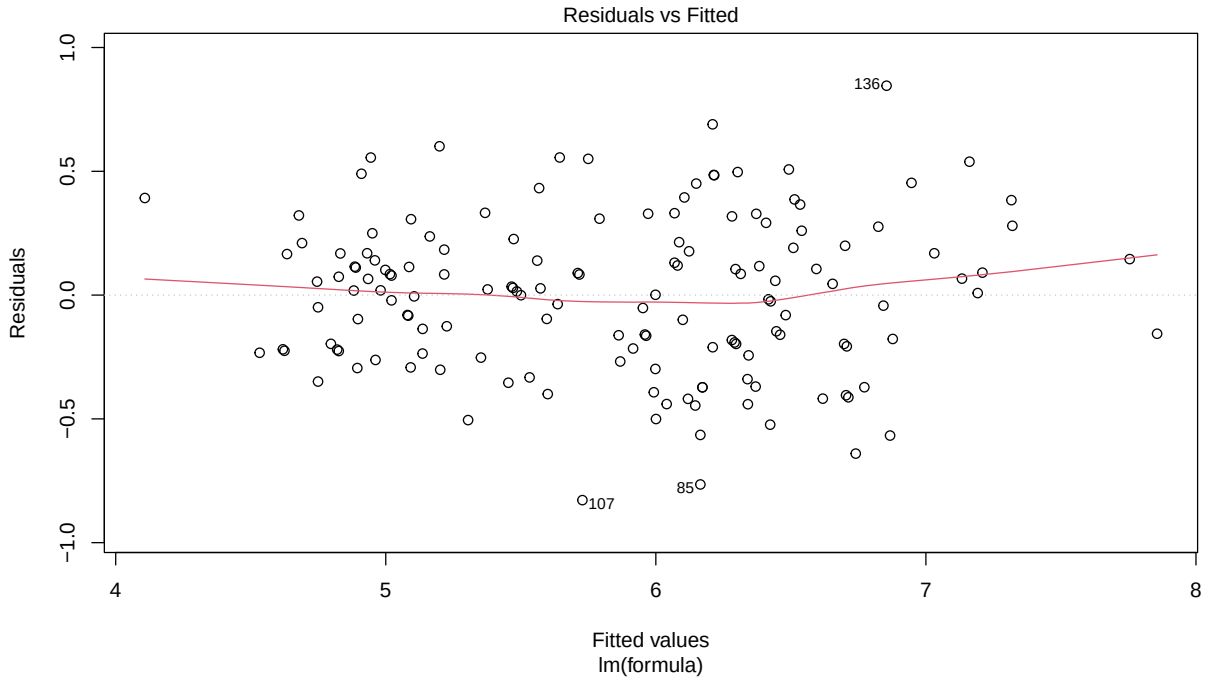
```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
dwtest(model)
```

```
##
##  Durbin-Watson test
##
## data:  model
## DW = 2.0604, p-value = 0.6013
## alternative hypothesis: true autocorrelation is greater than 0
```

단계 2-2: 등분산성 검정 - 잔차와 적합값의 분포

```
plot(model, which = 1)
```



단계 2-3: 잔차의 정규성 검정

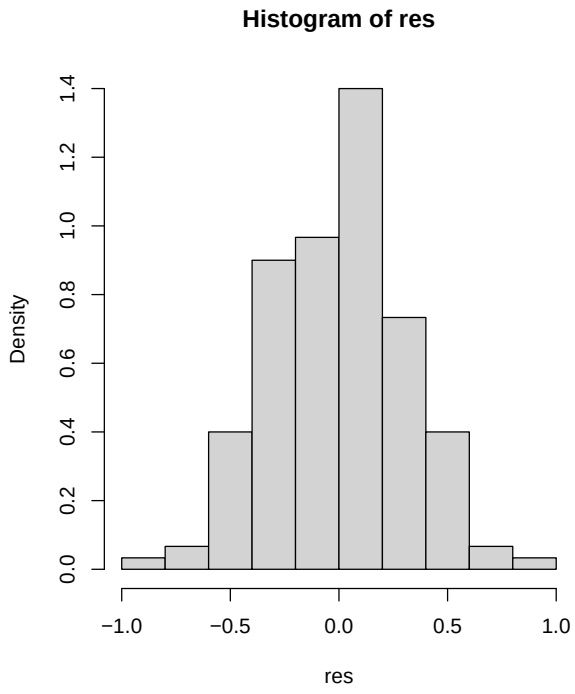
```
attributes(model)
```

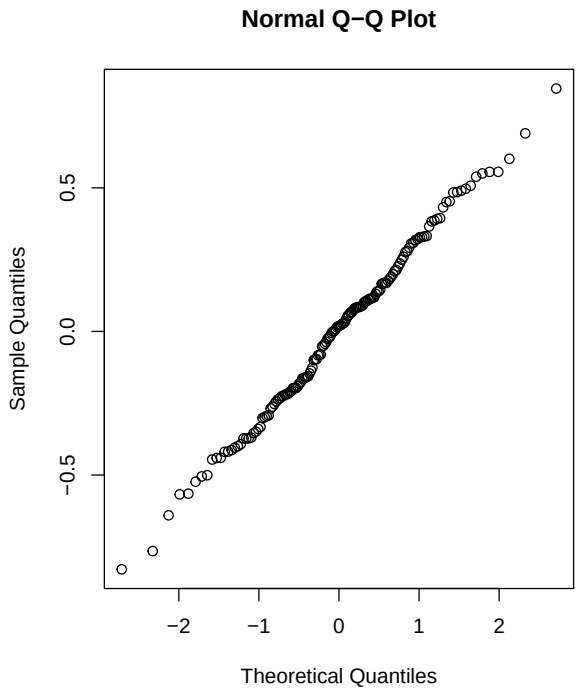
```
## $names
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
##
## $class
## [1] "lm"
```

```
res <- residuals(model)
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.99559, p-value = 0.9349
```

```
par(mfrow = c(1, 2))
hist(res, freq = F)
qqnorm(res)
```





단계 3: 다중 공선성 검사

```
library(car)
sqrt(vif(model)) > 2
```

```
##  Sepal.Width Petal.Length  Petal.Width
##        FALSE         TRUE         TRUE
```

단계 4: 회귀모델 생성과 평가

```
formula = Sepal.Length ~ Sepal.Width + Petal.Length
model <- lm(formula = formula, data = iris)
summary(model)
```

```
##
## Call:
## lm(formula = formula, data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.96159 -0.23489  0.00077  0.21453  0.78557
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.24914    0.24797    9.07 7.04e-16 ***
## Sepal.Width   0.59552    0.06933    8.59 1.16e-14 ***
## Petal.Length  0.47192    0.01712   27.57  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.3333 on 147 degrees of freedom
## Multiple R-squared:  0.8402, Adjusted R-squared:  0.838
## F-statistic: 386.4 on 2 and 147 DF,  p-value: < 2.2e-16
```

날씨 관련 요인 변수로 비(rain) 유뮤 예측

단계 1: 데이터 가져오기

```
weather = read.csv("data/weather.csv", stringsAsFactors = F)
dim(weather)
```

```
## [1] 366  15
```

```
head(weather)
```

```
##         Date MinTemp MaxTemp Rainfall Sunshine WindGustDir WindGustSpeed
## 1 2014-11-01     8.0    24.3      0.0      6.3          NW            30
## 2 2014-11-02    14.0    26.9      3.6      9.7         ENE            39
## 3 2014-11-03    13.7    23.4      3.6      3.3          NW            85
## 4 2014-11-04    13.3    15.5     39.8      9.1          NW            54
## 5 2014-11-05     7.6    16.1      2.8     10.6         SSE            50
## 6 2014-11-06     6.2    16.9      0.0      8.2          SE            44
##   WindDir WindSpeed Humidity Pressure Cloud Temp RainToday RainTomorrow
## 1      NW        20       29   1015.0     7 23.6        No          Yes
## 2       W        17       36   1008.4     3 25.7       Yes          Yes
## 3     NNE         6       69   1007.2     7 20.2       Yes          Yes
## 4       W        24       56   1007.0     7 14.1       Yes          Yes
## 5     ESE        28       49   1018.5     7 15.4       Yes           No
## 6       E        24       57   1021.7     5 14.8        No           No
```

```
str(weather)
```

```
## 'data.frame':    366 obs. of  15 variables:
##  $ Date         : chr  "2014-11-01" "2014-11-02" "2014-11-03" "2014-11-04" ...
##  $ MinTemp      : num  8 14 13.7 13.3 7.6 6.2 6.1 8.3 8.8 8.4 ...
##  $ MaxTemp      : num  24.3 26.9 23.4 15.5 16.1 16.9 18.2 17 19.5 22.8 ...
##  $ Rainfall     : num  0 3.6 3.6 39.8 2.8 0 0.2 0 0 16.2 ...
##  $ Sunshine     : num  6.3 9.7 3.3 9.1 10.6 8.2 8.4 4.6 4.1 7.7 ...
##  $ WindGustDir  : chr  "NW" "ENE" "NW" "NW" ...
##  $ WindGustSpeed: int  30 39 85 54 50 44 43 41 48 31 ...
##  $ WindDir      : chr  "NW" "W" "NNE" "W" ...
##  $ WindSpeed    : int  20 17 6 24 28 24 26 24 17 6 ...
##  $ Humidity     : int  29 36 69 56 49 57 47 57 48 32 ...
##  $ Pressure     : num  1015 1008 1007 1007 1018 ...
##  $ Cloud        : int  7 3 7 7 7 5 6 7 7 1 ...
##  $ Temp         : num  23.6 25.7 20.2 14.1 15.4 14.8 17.3 15.5 18.9 21.7 ...
##  $ RainToday    : chr  "No" "Yes" "Yes" "Yes" ...
##  $ RainTomorrow : chr  "Yes" "Yes" "Yes" "Yes" ...
```

단계 2: 변수 선택과 더미 변수 생성

```
weather_df <- weather[ , c(-1, -6, -8, -14)]
str(weather_df)
```

```
## 'data.frame':    366 obs. of  11 variables:
##  $ MinTemp      : num  8 14 13.7 13.3 7.6 6.2 6.1 8.3 8.8 8.4 ...
##  $ MaxTemp      : num  24.3 26.9 23.4 15.5 16.1 16.9 18.2 17 19.5 22.8 ...
##  $ Rainfall     : num  0 3.6 3.6 39.8 2.8 0 0.2 0 0 16.2 ...
##  $ Sunshine     : num  6.3 9.7 3.3 9.1 10.6 8.2 8.4 4.6 4.1 7.7 ...
##  $ WindGustSpeed: int  30 39 85 54 50 44 43 41 48 31 ...
##  $ WindSpeed    : int  20 17 6 24 28 24 26 24 17 6 ...
##  $ Humidity     : int  29 36 69 56 49 57 47 57 48 32 ...
##  $ Pressure     : num  1015 1008 1007 1007 1018 ...
##  $ Cloud        : int  7 3 7 7 7 5 6 7 7 1 ...
##  $ Temp         : num  23.6 25.7 20.2 14.1 15.4 14.8 17.3 15.5 18.9 21.7 ...
##  $ RainTomorrow : chr  "Yes" "Yes" "Yes" "Yes" ...
```

```
weather_df$RainTomorrow[weather_df$RainTomorrow == 'Yes'] <- 1
weather_df$RainTomorrow[weather_df$RainTomorrow == 'No'] <- 0
weather_df$RainTomorrow <- as.numeric(weather_df$RainTomorrow)
head(weather_df)
```

```
##   MinTemp MaxTemp Rainfall Sunshine WindGustSpeed WindSpeed Humidity Pressure
## 1     8.0    24.3      0.0      6.3            30        20       29   1015.0
## 2    14.0    26.9      3.6      9.7            39        17       36   1008.4
## 3    13.7    23.4      3.6      3.3            85         6       69   1007.2
## 4    13.3    15.5     39.8      9.1            54        24       56   1007.0
## 5     7.6    16.1      2.8     10.6            50        28       49   1018.5
## 6     6.2    16.9      0.0      8.2            44        24       57   1021.7
##   Cloud Temp RainTomorrow
## 1     7 23.6            1
## 2     3 25.7            1
## 3     7 20.2            1
## 4     7 14.1            1
## 5     7 15.4            0
## 6     5 14.8            0
```

단계 3: 학습데이터와 검정데이터 생성(7:3 비율)

```
idx <- sample(1:nrow(weather_df), nrow(weather_df) * 0.7)
train <- weather_df[idx, ]
test <- weather_df[-idx, ]
```

단계 4: 로지스틱 회귀모델 생성

```
weather_model <- glm(RainTomorrow ~ ., data = train, family = 'binomial')
weather_model
```

```
##
## Call:  glm(formula = RainTomorrow ~ ., family = "binomial", data = train)
##
## Coefficients:
##   (Intercept)        MinTemp        MaxTemp       Rainfall       Sunshine
##     110.54207       -0.21551        0.25377        0.07239       -0.35173
## WindGustSpeed      WindSpeed       Humidity       Pressure          Cloud
##       0.05896       -0.01916        0.08956       -0.12127        0.15745
##          Temp
##       0.12919
```

```
##
## Degrees of Freedom: 251 Total (i.e. Null);  241 Residual
##   (4 observations deleted due to missingness)
## Null Deviance:       242.5
## Residual Deviance: 121.9     AIC: 143.9
```

```
summary(weather_model)
```

```
##
## Call:
## glm(formula = RainTomorrow ~ ., family = "binomial", data = train)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   110.54207   50.00194   2.211  0.02705 *
## MinTemp        -0.21551    0.09311  -2.315  0.02063 *
## MaxTemp         0.25377    0.23998   1.057  0.29031
## Rainfall        0.07239    0.05170   1.400  0.16145
## Sunshine       -0.35173    0.13382  -2.628  0.00858 **
## WindGustSpeed   0.05896    0.02931   2.012  0.04427 *
## WindSpeed      -0.01916    0.04262  -0.449  0.65310
## Humidity        0.08956    0.03126   2.865  0.00417 **
## Pressure       -0.12127    0.04826  -2.513  0.01198 *
## Cloud           0.15745    0.13799   1.141  0.25388
## Temp            0.12919    0.24412   0.529  0.59666
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 242.48  on 251  degrees of freedom
## Residual deviance: 121.86  on 241  degrees of freedom
##   (4 observations deleted due to missingness)
## AIC: 143.86
##
## Number of Fisher Scoring iterations: 6
```

단계 5: 로지스틱 회귀모델 예측치 생성

```
pred <- predict(weather_model, newdata = test, type = "response")
pred[is.na(pred)] <- 0
pred
```

```
##            1            2           12           15           16           23
## 0.1006023380 0.1088883965 0.0343724929 0.0239357285 0.1460033621 0.9507757900
##           29           31           35           36           39           40
## 0.9400093629 0.9890821356 0.0384781108 0.4057386961 0.0818908277 0.0428951877
##           53           63           64           71           72           73
## 0.2434585661 0.1566856281 0.6049490674 0.0454415094 0.2452495242 0.7047456968
##           76           78           79           81           85           86
## 0.1079822080 0.0178549330 0.4666856479 0.9035742149 0.0583250715 0.0382410806
##           87           88           91           95           96           97
## 0.0861191078 0.1753407237 0.6767897874 0.2840203721 0.9194196354 0.9520043114
##           98          100          107          108          112          113
## 0.8396805620 0.7288952241 0.0065004492 0.0108436310 0.4325155848 0.8647898498
```

```
##          115          119          121          125          127          128
## 0.0238210422 0.2140467774 0.0062359870 0.2203071609 0.0467568766 0.5168297891
##          133          139          140          141          142          145
## 0.0646656135 0.0433665717 0.0317397724 0.5714360448 0.1123262924 0.5804536269
##          148          152          156          157          158          162
## 0.0106245078 0.0270971136 0.0056797290 0.0030158748 0.0147270563 0.0055354255
##          172          184          187          190          192          193
## 0.0544577739 0.0126530657 0.1143166706 0.0152181742 0.0196135097 0.0890375055
##          200          201          204          207          213          215
## 0.0575666911 0.0818863051 0.0008647838 0.0755508621 0.0073428705 0.2039002461
##          218          219          220          224          228          234
## 0.1102371825 0.0023391909 0.0025655532 0.3653374596 0.0135465030 0.0655518299
##          235          238          245          246          248          249
## 0.0026896961 0.0034935536 0.0201832088 0.0041838934 0.0041640072 0.0515383893
##          255          260          261          268          272          273
## 0.0160322812 0.1204974445 0.2754022893 0.0010848743 0.0000000000 0.0123610167
##          274          279          282          294          297          304
## 0.1175763498 0.4318834729 0.0240842121 0.0175756166 0.0010426445 0.3091961411
##          312          313          314          315          319          321
## 0.0352824279 0.0053895692 0.0015182046 0.0059684054 0.2639932751 0.0023464413
##          325          336          341          343          344          346
## 0.0115531163 0.0048167162 0.0647745358 0.0059949730 0.0014537220 0.0018005695
##          347          352          354          356          360          361
## 0.0075969707 0.0213471127 0.1479056633 0.0161954574 0.0369391249 0.4511026959
##          362          364
## 0.2258150735 0.0228924180
```

시그모이드 함수

```
result_pred <- ifelse(pred >= 0.5, 1, 0)
result_pred
```

```
##   1   2  12  15  16  23  29  31  35  36  39  40  53  63  64  71  72  73  76  78
##   0   0   0   0   0   1   1   1   0   0   0   0   0   0   1   0   0   1   0   0
##  79  81  85  86  87  88  91  95  96  97  98 100 107 108 112 113 115 119 121 125
##   0   1   0   0   0   0   1   0   1   1   1   1   0   0   0   1   0   0   0   0
## 127 128 133 139 140 141 142 145 148 152 156 157 158 162 172 184 187 190 192 193
##   0   1   0   0   0   1   0   1   0   0   0   0   0   0   0   0   0   0   0   0
## 200 201 204 207 213 215 218 219 220 224 228 234 235 238 245 246 248 249 255 260
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 261 268 272 273 274 279 282 294 297 304 312 313 314 315 319 321 325 336 341 343
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 344 346 347 352 354 356 360 361 362 364
##   0   0   0   0   0   0   0   0   0   0
```

```
table(result_pred)
```

```
## result_pred
##  0  1
## 95 15
```

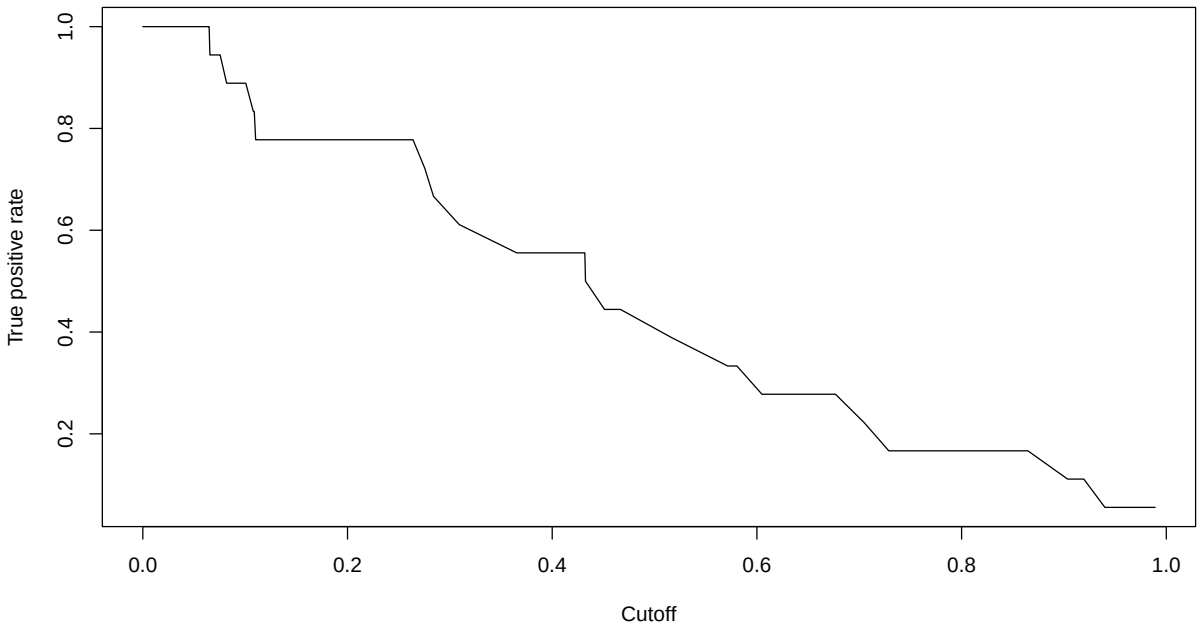
단계 6: 모델 평가 - 분류정확도 계산

```
table(result_pred, test$RainTomorrow)
```

```
##
## result_pred  0  1
##           0 84 11
##           1  8  7
```

단계 7: ROC Curve를 이용한 모델 평가

```
# install.packages("ROCR")
library(ROCR)
pr <- prediction(pred, test$RainTomorrow)
prf <- performance(pr, measure = "tpr", x.maeasure = "fpr")
plot(prf)
```



의사결정 트리 생성: ctree() 함수 이용

단계 1: party 패키지 설치

```
# install.packages("party")
library(party)
```

```
## Loading required package: grid
## Loading required package: mvtnorm
## Loading required package: modeltools
## Loading required package: stats4
##
## Attaching package: 'modeltools'
```

```
## The following object is masked from 'package:car':
##
##     Predict
```

```
## Loading required package: strucchange
```

```
## Loading required package: sandwich
```

단계 2: airquality 데이터 셋 로딩

```
#install.packages("datasets")
library(datasets)
str(airquality)
```

```
## 'data.frame':    153 obs. of  6 variables:
##  $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
##  $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
##  $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
##  $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
##  $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

단계 3: formula 생성

```
formula <- Temp ~ Solar.R + Wind + Ozone
```

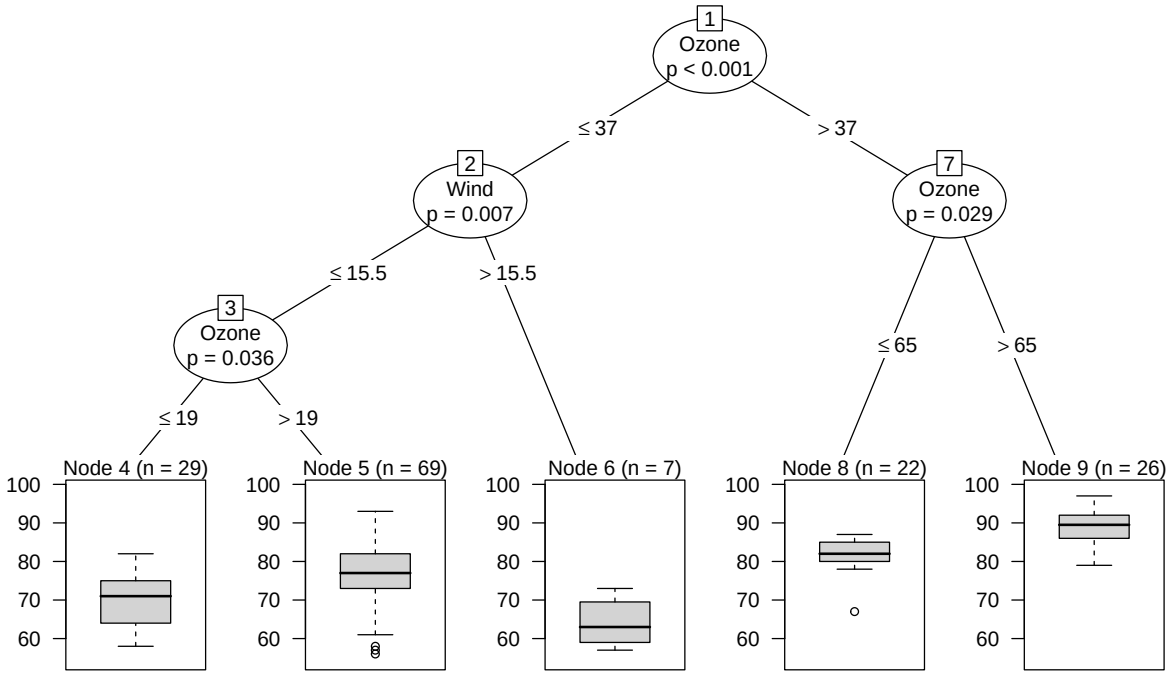
단계 4: 분류모델 생성 - formula를 이용하여 분류모델 생성

```
air_ctree <- ctree(formula, data = airquality)
air_ctree
```

```
##
##   Conditional inference tree with 5 terminal nodes
##
## Response:  Temp
## Inputs:  Solar.R, Wind, Ozone
## Number of observations:  153
##
## 1) Ozone <= 37; criterion = 1, statistic = 56.086
##   2) Wind <= 15.5; criterion = 0.993, statistic = 9.387
##     3) Ozone <= 19; criterion = 0.964, statistic = 6.299
##       4)*  weights = 29
##     3) Ozone > 19
##       5)*  weights = 69
##   2) Wind > 15.5
##     6)*  weights = 7
## 1) Ozone > 37
##   7) Ozone <= 65; criterion = 0.971, statistic = 6.691
##     8)*  weights = 22
##   7) Ozone > 65
##     9)*  weights = 26
```

단계 5: 분류분석 결과

```
plot(air_ctree)
```



학습데이터와 검정데이터 샘플링으로 분류분석 수행

단계 1: 학습데이터와 검정데이터 샘플링

```
set.seed(42)
idx <- sample(1:nrow(iris), nrow(iris) * 0.7)
train <- iris[idx, ]
test <- iris[-idx, ]
```

단계 2: formula(공식) 생성

```
formula <- Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
```

단계 3: 학습데이터 이용 분류모델 생성

```
iris_ctree <- ctree(formula, data = train)
iris_ctree
```

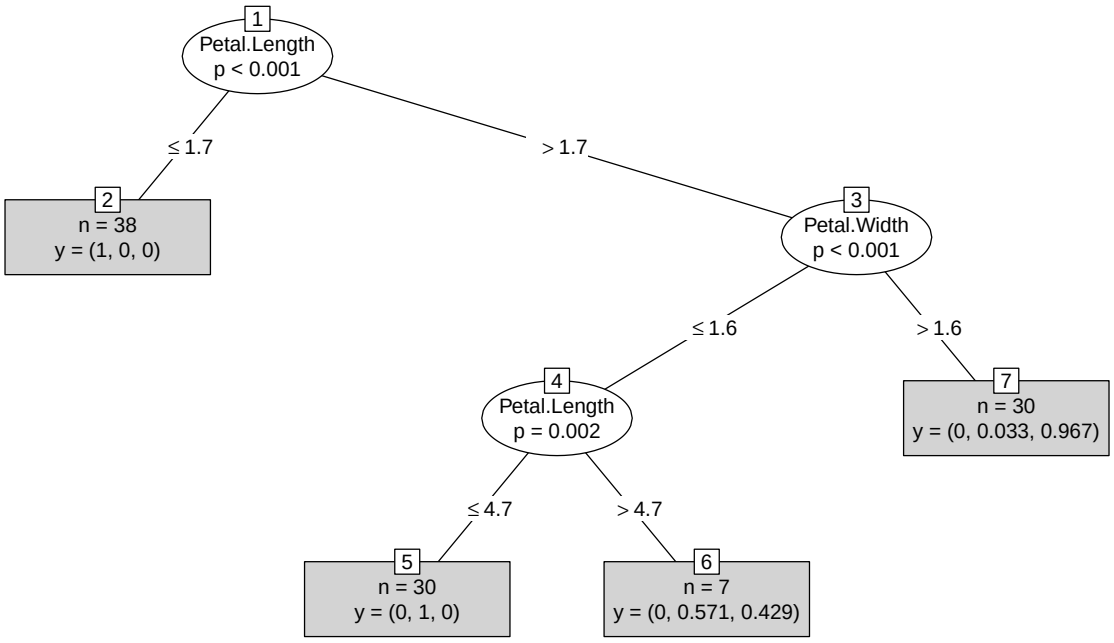
```
##
##   Conditional inference tree with 4 terminal nodes
##
## Response:  Species
## Inputs:  Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
## Number of observations:  105
```

```
##
## 1) Petal.Length <= 1.7; criterion = 1, statistic = 97.909
##   2)*  weights = 38
## 1) Petal.Length > 1.7
##   3) Petal.Width <= 1.6; criterion = 1, statistic = 44.442
##     4) Petal.Length <= 4.7; criterion = 0.998, statistic = 11.769
##       5)*  weights = 30
##     4) Petal.Length > 4.7
##       6)*  weights = 7
##   3) Petal.Width > 1.6
##     7)*  weights = 30
```

단계 4: 분류모델 플로팅

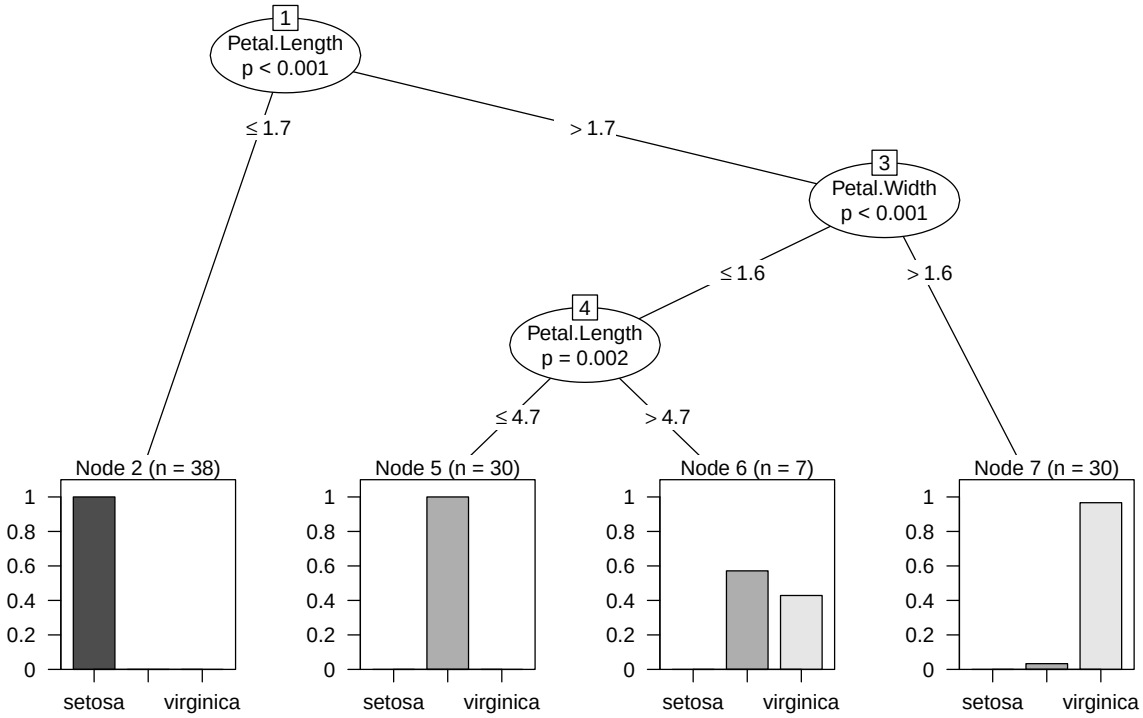
단계 4-1: 간단한 형식으로 시각화

```
plot(iris_ctree, type = "simple")
```



단계 4-2: 의사결정 트리로 플로팅

```
plot(iris_ctree)
```



단계 5: 분류모델 평가

단계 5-1: 모델의 예측치 생성과 혼돈 매트릭스 생성

```
pred <- predict(iris_ctree, test)
table(pred, test$Species)
```

```
##
## pred         setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      2         14         1
##   virginica       0          1        17
```

단계 5-2: 분류 정확도 - 96%

```
(14 + 16 + 13) / nrow(test)
```

```
## [1] 0.9555556
```

K겹 교차 검정 샘플링으로 분류 분석하기

단계 1: K겹 교차 검정을 위한 샘플링 - 3겹, 2회 반복

```
# install.packages("cvTools")
library(cvTools)
```

```
## Loading required package: lattice
```

```
## Loading required package: robustbase
```

```
cross <- cvFolds(nrow(iris), K = 3, R = 2)
```

단계 2: K겹 교차 검정 데잍 보기

```
str(cross)
```

```
## List of 5
##  $ n      : num 150
##  $ K      : num 3
##  $ R      : num 2
##  $ subsets: int [1:150, 1:2] 43 42 149 97 25 115 32 81 14 111 ...
##  $ which  : int [1:150] 1 2 3 1 2 3 1 2 3 1 ...
##  - attr(*, "class")= chr "cvFolds"
```

```
cross
```

```
##
## Repeated 3-fold CV with 2 replications:
## Fold      1   2
##    1     43 115
##    2     42 105
##    3    149  11
##    1     97 147
##    2     25  36
##    3    115  59
##    1     32 113
##    2     81  30
##    3     14 108
##    1    111  47
##    2      6 149
##    3    150  88
##    1    113 114
##    2    138 135
##    3     31  12
##    1     94  51
##    2    140  10
##    3     38 121
##    1     95  89
##    2    134  37
##    3     84  27
##    1     15  46
##    2     34  31
##    3    148  91
##    1     87  80
##    2     60 100
##    3     12 120
##    1     26  53
##    2     41  24
##    3     65  38
##    1     66 116
##    2     56  85
##    3     24  20
##    1     98  75
##    2    146  97
```

```
##     3     92  78
##     1    107  92
##     2     61  33
##     3     62  93
##     1    142 146
##     2    128  45
##     3    120  96
##     1    132 102
##     2    144  22
##     3     27  44
##     1     10  64
##     2     57   7
##     3     28   5
##     1     37 128
##     2    105  32
##     3      5  82
##     1     35  39
##     2     78  13
##     3    141 143
##     1    103  54
##     2     54   2
##     3     90  28
##     1    136   3
##     2    139 144
##     3     52  84
##     1    143   9
##     2     59 109
##     3    106 134
##     1     30  71
##     2     91 125
##     3    130  25
##     1     75  74
##     2    135  73
##     3     73  19
##     1     17  79
##     2    112 129
##     3     13 112
##     1     63 107
##     2     49  61
##     3    137 118
##     1      1  42
##     2     83  17
##     3     33 126
##     1     96 127
##     2      2 131
##     3     93 110
##     1      8  50
##     2      3 137
##     3     79  48
##     1    124  34
##     2    121  58
##     3    126   4
##     1     51 119
##     2    125  26
```

```
##     3     70  77
##     1    102  99
##     2     47 138
##     3    119  83
##     1    131  52
##     2     55 150
##     3     16  68
##     1    101 106
##     2      7  86
##     3      9   6
##     1    122 132
##     2    116  63
##     3     68 124
##     1     18 117
##     2     89  60
##     3     72  69
##     1    129  95
##     2     29 133
##     3     20  90
##     1     50 148
##     2     58  57
##     3    118 142
##     1     82 103
##     2    133  14
##     3     22   1
##     1    123 139
##     2    145 136
##     3     45  56
##     1     21  87
##     2     71  67
##     3    127  65
##     1    104 140
##     2     85  81
##     3    147 130
##     1    117  21
##     2     74  41
##     3     11 141
##     1     48 111
##     2     23  76
##     3    114  70
##     1    108  49
##     2      4  23
##     3    110  43
##     1     53  55
##     2     36  18
##     3     76  29
##     1     46  15
##     2    100  66
##     3     64 122
##     1     99 145
##     2     80   8
##     3     39  94
##     1     86  40
##     2     77 123
```

```
##    3    109  62
##    1     67 101
##    2     88  98
##    3     44  16
##    1     69  35
##    2     40  72
##    3     19 104
```

```
length(cross$which)
```

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

```
dim(cross$subsets)
```

```
## [1] 150   2
```

```
table(cross$which)
```

```
##
##  1  2  3
## 50 50 50
```

단계 3: K겹 교차 검정 수행

```
R = 1:2
K = 1:3
CNT = 0
ACC <- numeric()

for(r in R) {
  cat('\n R = ', r, '\n')
  for(k in K) {

    datas_ids <- cross$subsets[cross$which == k, r]
    test <- iris[datas_ids, ]
    cat('test : ', nrow(test), '\n')

    formual <- Species ~ .
    train <- iris[-datas_ids, ]
    cat('train : ', nrow(train), '\n')

    model <- ctree(Species ~ ., data = train)
    pred <- predict(model, test)
    t <- table(pred, test$Species)
    print(t)

    CNT <- CNT + 1
    ACC[CNT] <- (t[1, 1] + t[2, 2] + t[3, 3]) / sum(t)
  }
}
```

```
##
##  R =  1
## test :  50
## train :  100
##
```

```
## pred         setosa versicolor virginica
##   setosa         16          0         0
##   versicolor      0         15         1
##   virginica       0          1        17
## test :  50
## train :  100
##
## pred         setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         18         2
##   virginica       0          2        13
## test :  50
## train :  100
##
## pred         setosa versicolor virginica
##   setosa         19          0         0
##   versicolor      0         14         2
##   virginica       0          0        15
##
##  R =  2
## test :  50
## train :  100
##
## pred         setosa versicolor virginica
##   setosa         13          0         0
##   versicolor      0         15         3
##   virginica       0          1        18
## test :  50
## train :  100
##
## pred         setosa versicolor virginica
##   setosa         20          0         0
##   versicolor      0         16         1
##   virginica       0          0        13
## test :  50
## train :  100
##
## pred         setosa versicolor virginica
##   setosa         17          0         0
##   versicolor      0         17         3
##   virginica       0          1        12
```

```
CNT
```

```
## [1] 6
```

단계 4: 교차 검정 모델 평가

```
ACC
```

```
## [1] 0.96 0.92 0.96 0.92 0.98 0.92
```

```
length(ACC)
```

```
## [1] 6
```

```
result_acc <- mean(ACC, na.rm = T)
result_acc
```

```
## [1] 0.9433333
```

고속도로 주행거리에 미치는 영향변수 보기

단계 1: 패키지 설치 및 로딩

```
# install.packages("tidyverse")
library(ggplot2)
data(mpg)
```

단계 2: 학습데이터와 검정데이터 생성

```
t <- sample(1:nrow(mpg), 120)
train <- mpg[-t, ]
test <- mpg[t, ]
dim(train)
```

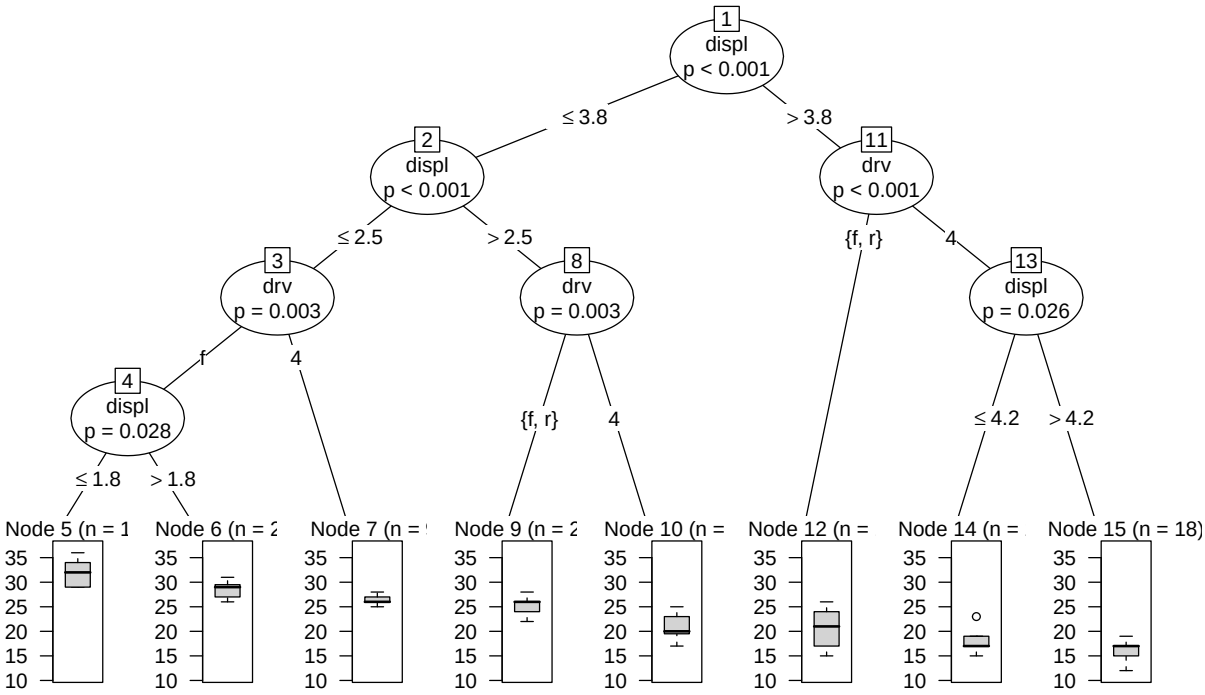
```
## [1] 114  11
```

```
dim(test)
```

```
## [1] 120  11
```

단계 3: formula 작성과 분류모델 생성

```
test$drv <- factor(test$drv)
formula <- hwy ~ displ + cyl + drv
tree_model <- ctree(formula, data = test)
plot(tree_model)
```



AdultUCI 데잍 셋을 이용한 분류분석

단계 1: 패키지 설치 및 데이터 셋 구조 보기

```
# install.packages("arules")
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following object is masked from 'package:modeltools':
##
##     info
```

```
## The following object is masked from 'package:car':
##
##     recode
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
data(AdultUCI)
str(AdultUCI)
```

```
## 'data.frame':    48842 obs. of  15 variables:
##  $ age           : int  39 50 38 53 28 37 49 52 31 42 ...
##  $ workclass     : Factor w/ 8 levels "Federal-gov",..: 7 6 4 4 4 4 4 6 4 4 ...
##  $ fnlwgt        : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 159449 ...
##  $ education     : Ord.factor w/ 16 levels "Preschool"<"1st-4th"<..: 14 14 9 7 14 15 5 9 15 14 ...
```

```
##  $ education-num : int  13 13 9 7 13 14 5 9 14 13 ...
##  $ marital-status: Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 5 3 1 3 3 3 4 3 5 3 ...
##  $ occupation    : Factor w/ 14 levels "Adm-clerical",..: 1 4 6 6 10 4 8 4 10 4 ...
##  $ relationship  : Factor w/ 6 levels "Husband","Not-in-family",..: 2 1 2 1 6 6 2 1 2 1 ...
##  $ race          : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 3 3 5 3 5 5 5 ...
##  $ sex           : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 1 2 1 2 ...
##  $ capital-gain  : int  2174 0 0 0 0 0 0 0 14084 5178 ...
##  $ capital-loss  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hours-per-week: int  40 13 40 40 40 40 16 45 50 40 ...
##  $ native-country: Factor w/ 41 levels "Cambodia","Canada",..: 39 39 39 39 5 39 23 39 39 39 ...
##  $ income        : Ord.factor w/ 2 levels "small"<"large": 1 1 1 1 1 1 1 2 2 2 ...
```

```
names(AdultUCI)
```

```
##  [1] "age"            "workclass"      "fnlwgt"         "education"
##  [5] "education-num"  "marital-status" "occupation"     "relationship"
##  [9] "race"           "sex"            "capital-gain"   "capital-loss"
## [13] "hours-per-week" "native-country" "income"
```

단계 2: 데이터 샘플링 - 10,000개 관측치]선택

```
set.seed(1234)
choice <- sample(1:nrow(AdultUCI), 10000)
adult.df <- AdultUCI[choice, ]
str(adult.df)
```

```
## 'data.frame':    10000 obs. of  15 variables:
##  $ age           : int  76 34 44 44 50 36 17 26 43 25 ...
##  $ workclass     : Factor w/ 8 levels "Federal-gov",..: 6 6 4 4 4 4 4 4 4 4 ...
##  $ fnlwgt        : int  106430 201292 318046 368757 115284 207853 158704 147821 160246 135645 ...
##  $ education     : Ord.factor w/ 16 levels "Preschool"<"1st-4th"<..: 5 12 13 13 15 14 6 14 13 15 ...
##  $ education-num : int  5 11 10 10 14 13 6 13 10 14 ...
##  $ marital-status: Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 3 3 3 3 3 3 5 5 1 5 ...
##  $ occupation    : Factor w/ 14 levels "Adm-clerical",..: 5 5 12 7 3 12 12 12 10 12 ...
##  $ relationship  : Factor w/ 6 levels "Husband","Not-in-family",..: 1 1 1 1 1 1 4 4 5 2 ...
##  $ race          : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 5 5 5 5 5 3 5 ...
##  $ sex           : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 1 1 2 ...
##  $ capital-gain  : int  0 0 0 0 0 15024 0 0 0 0 ...
##  $ capital-loss  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hours-per-week: int  40 50 35 40 40 65 20 45 40 20 ...
##  $ native-country: Factor w/ 41 levels "Cambodia","Canada",..: 39 39 39 39 39 39 39 NA 39 39 ...
##  $ income        : Ord.factor w/ 2 levels "small"<"large": NA NA NA 1 NA NA 1 1 1 1 ...
```

단계 3: 변수 추출 및 데이터프레임 생성

단계 3-1: 변수 추출

```
capital <- adult.df$`capital-gain`
hours <- adult.df$`hours-per-week`
education <- adult.df$`education-num`
race <- adult.df$race
age <- adult.df$age
income <- adult.df$income
```

단계 3-2: 데이터프레임 생성

```
adult_df <- data.frame(capital = capital, age = age, race = race,
                       hours = hours, education = education, income = income)
str(adult_df)
```

```
## 'data.frame':    10000 obs. of  6 variables:
##  $ capital  : int  0 0 0 0 0 15024 0 0 0 0 ...
##  $ age      : int  76 34 44 44 50 36 17 26 43 25 ...
##  $ race     : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 5 5 5 5 5 3 5 ...
##  $ hours    : int  40 50 35 40 40 65 20 45 40 20 ...
##  $ education: int  5 11 10 10 14 13 6 13 10 14 ...
##  $ income   : Ord.factor w/ 2 levels "small"<"large": NA NA NA 1 NA NA 1 1 1 1 ...
```

단계 4: formula 생성 - 자본이득(capital)에 영향을 미치는 변수

```
formula <- capital ~ income + education + hours + race + age
```

단계 5: 분류모델 생성 및 예측

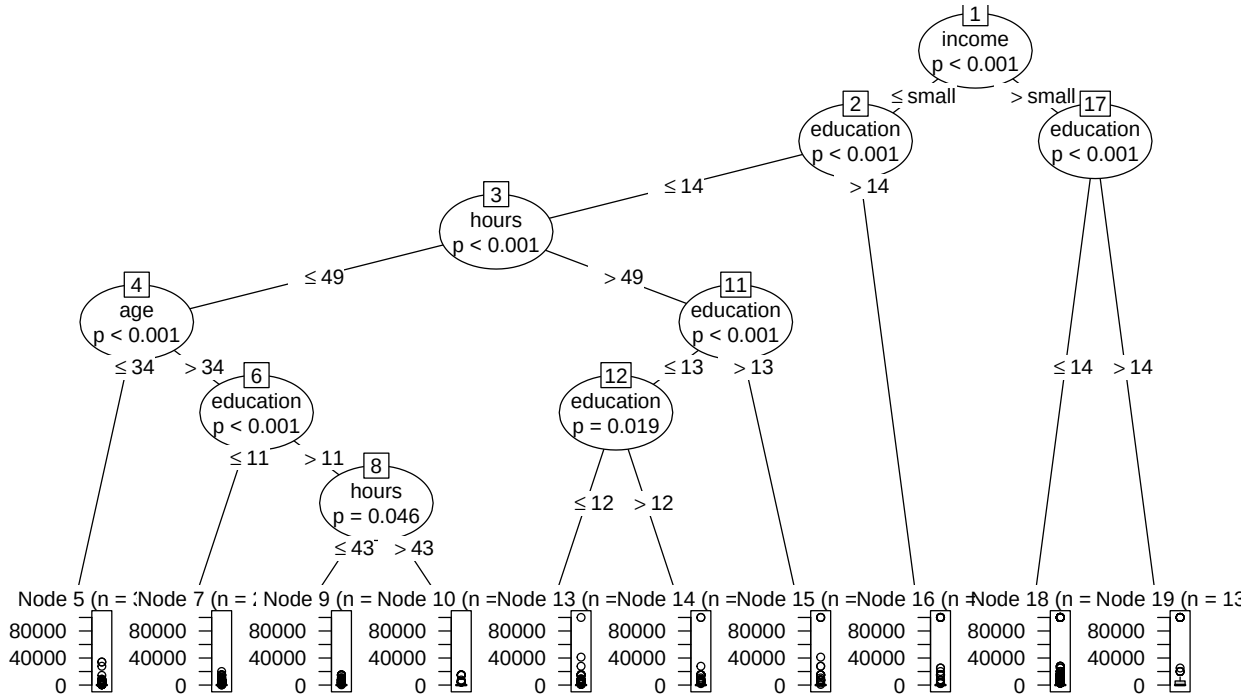
```
adult_ctree <- ctree(formula, data = adult_df)
adult_ctree
```

```
##
##   Conditional inference tree with 10 terminal nodes
##
## Response:  capital
## Inputs:  income, education, hours, race, age
## Number of observations:  10000
##
## 1) income <= small; criterion = 1, statistic = 324.284
##   2) education <= 14; criterion = 1, statistic = 96.537
##     3) hours <= 49; criterion = 1, statistic = 54.964
##       4) age <= 34; criterion = 1, statistic = 54.112
##         5)*  weights = 3469
##       4) age > 34
##         6) education <= 11; criterion = 1, statistic = 29.521
##           7)*  weights = 2757
##         6) education > 11
##           8) hours <= 43; criterion = 0.954, statistic = 6.762
##             9)*  weights = 640
##           8) hours > 43
##             10)*  weights = 101
##     3) hours > 49
##       11) education <= 13; criterion = 1, statistic = 19.25
##         12) education <= 12; criterion = 0.981, statistic = 8.355
##           13)*  weights = 931
##         12) education > 12
##           14)*  weights = 269
##       11) education > 13
##         15)*  weights = 117
##   2) education > 14
##     16)*  weights = 143
## 1) income > small
```

```
##   17) education <= 14; criterion = 1, statistic = 16.826
##     18)*  weights = 1434
##   17) education > 14
##     19)*  weights = 139
```

단계 6: 분류모델 시각화

```
plot(adult_ctree)
```



단계 7: 자본이득(capital) 요약통계량 보기

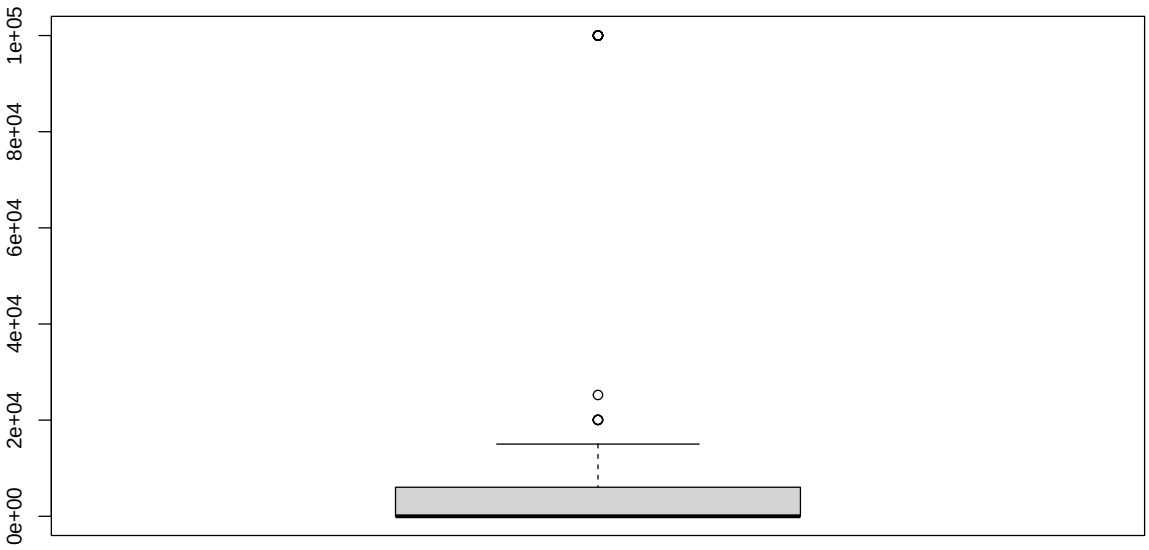
```
adultResult <- subset(adult_df,
                      adult_df$income == 'large' &
                        adult_df$education > 14)
length(adultResult$education)
```

```
## [1] 139
```

```
summary(adultResult$capital)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0       0    9241    6042   99999
```

```
boxplot(adultResult$capital)
```



rpart() 함수를 이용한 의사결정 트리 생성

단계 1: 패키지 설치 및 로딩

```
# install.packages("rpart")
# install.packages("rpart.plot")
library(rpart)
library(rpart.plot)
```

단계 2: 데잍 로딩

```
data(iris)
```

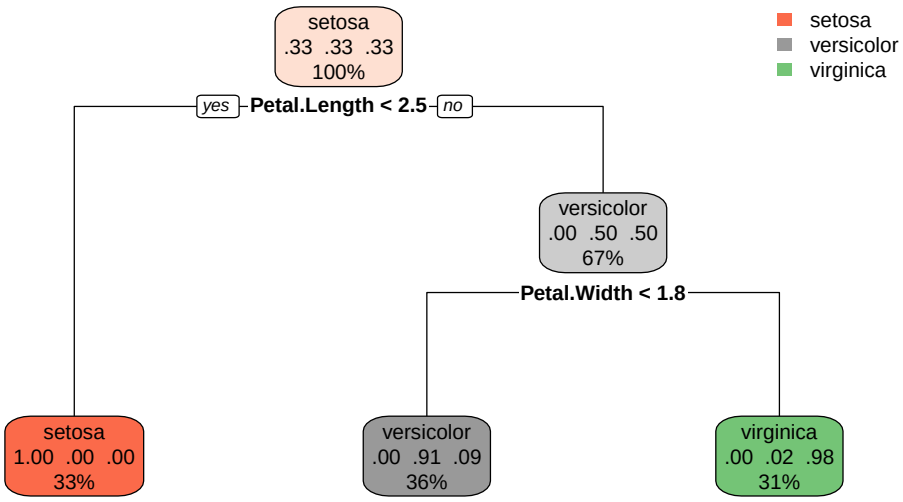
단계 3: rpart() 함수를 이용한 분류분석

```
rpart_model <- rpart(Species ~ ., data = iris)
rpart_model
```

```
## n= 150
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
##   2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *
##   3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)
##     6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259) *
##     7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
```

단계 4: 분류분석 시각화

```
rpart.plot(rpart_model)
```



날씨 데이터를 이용하여 비(rain) 유무 예측

단계 1: 데이터 가져오기

```
weather = read.csv("data/weather.csv", header = TRUE)
```

단계 2: 데이터 특성 보기

```
str(weather)
```

```
## 'data.frame':    366 obs. of  15 variables:
##  $ Date         : chr  "2014-11-01" "2014-11-02" "2014-11-03" "2014-11-04" ...
##  $ MinTemp      : num  8 14 13.7 13.3 7.6 6.2 6.1 8.3 8.8 8.4 ...
##  $ MaxTemp      : num  24.3 26.9 23.4 15.5 16.1 16.9 18.2 17 19.5 22.8 ...
##  $ Rainfall     : num  0 3.6 3.6 39.8 2.8 0 0.2 0 0 16.2 ...
##  $ Sunshine     : num  6.3 9.7 3.3 9.1 10.6 8.2 8.4 4.6 4.1 7.7 ...
##  $ WindGustDir  : chr  "NW" "ENE" "NW" "NW" ...
##  $ WindGustSpeed: int  30 39 85 54 50 44 43 41 48 31 ...
##  $ WindDir      : chr  "NW" "W" "NNE" "W" ...
##  $ WindSpeed    : int  20 17 6 24 28 24 26 24 17 6 ...
##  $ Humidity     : int  29 36 69 56 49 57 47 57 48 32 ...
##  $ Pressure     : num  1015 1008 1007 1007 1018 ...
##  $ Cloud        : int  7 3 7 7 7 5 6 7 7 1 ...
##  $ Temp         : num  23.6 25.7 20.2 14.1 15.4 14.8 17.3 15.5 18.9 21.7 ...
##  $ RainToday    : chr  "No" "Yes" "Yes" "Yes" ...
```

```
##  $ RainTomorrow : chr  "Yes" "Yes" "Yes" "Yes" ...
```

```
head(weather)
```

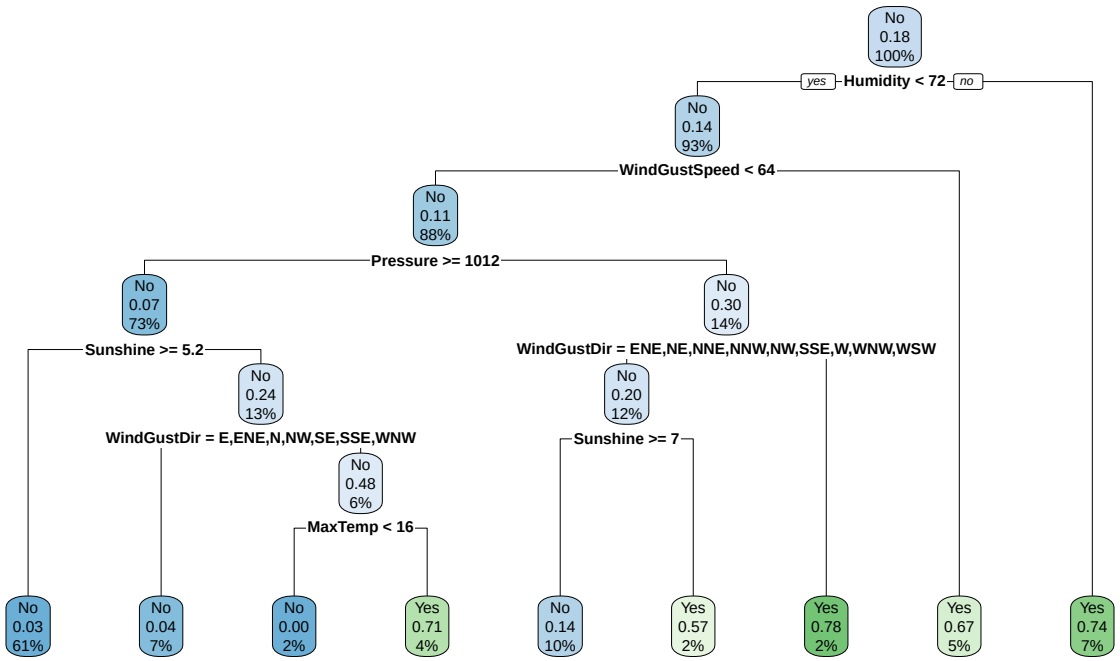
```
##         Date MinTemp MaxTemp Rainfall Sunshine WindGustDir WindGustSpeed
## 1 2014-11-01     8.0    24.3      0.0      6.3          NW             30
## 2 2014-11-02    14.0    26.9      3.6      9.7         ENE             39
## 3 2014-11-03    13.7    23.4      3.6      3.3          NW             85
## 4 2014-11-04    13.3    15.5     39.8      9.1          NW             54
## 5 2014-11-05     7.6    16.1      2.8     10.6         SSE             50
## 6 2014-11-06     6.2    16.9      0.0      8.2          SE             44
##   WindDir WindSpeed Humidity Pressure Cloud Temp RainToday RainTomorrow
## 1      NW        20       29   1015.0     7 23.6        No          Yes
## 2       W        17       36   1008.4     3 25.7       Yes          Yes
## 3     NNE         6       69   1007.2     7 20.2       Yes          Yes
## 4       W        24       56   1007.0     7 14.1       Yes          Yes
## 5     ESE        28       49   1018.5     7 15.4       Yes           No
## 6       E        24       57   1021.7     5 14.8        No           No
```

단계 3: 분류분석 데이터 가져오기

```
weather.df <- rpart(RainTomorrow ~ ., data = weather[ , c(-1, -14)], cp = 0.01)
```

단계 4: 분류분석 시각화

```
rpart.plot(weather.df)
```



단계 5: 예측치 생성과 코딩 변경

단계 5-1: 예측치 생성

```
weather_pred <- predict(weather.df, weather)
```

단계 5-2: y의 범주로 코딩 변환 - Yes(0.5이상), No(0.5미만)

```
weather_pred2 <- ifelse(weather_pred[ , 2] >= 0.5, 'Yes', 'No')
```

단계 6: 모델 평가

```
table(weather_pred2, weather$RainTomorrow)
```

```
##
## weather_pred2  No Yes
##           No  278  13
##           Yes  22  53
```

```
(278 + 53) / nrow(weather)
```

```
## [1] 0.9043716
```

랜덤 포레스트 기본 모델 생성

단계 1: 패키지 설치 및 데이터 셋 가져오기

```
# install.packages("randomForest")
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
data(iris)
```

단계 2: 랜덤 포레스트 모델 ㄹ생성

```
model <- randomForest(Species ~ ., data = iris)
```

파라미터 조정 - 트리 개수 300개, 변수 개수 4개 지정

```
model2 <- randomForest(Species ~ ., data = iris,
                       ntree = 300, mtry = 4, na.action = na.omit)
```

중요 변수를 생성하여 랜덤 포레스트 모델 생성

단계 1: 중요 변수로 랜덤 포레스트 모델 생성

```
model3 <- randomForest(Species ~ ., data = iris, importance = T, na.action = na.omit)
```

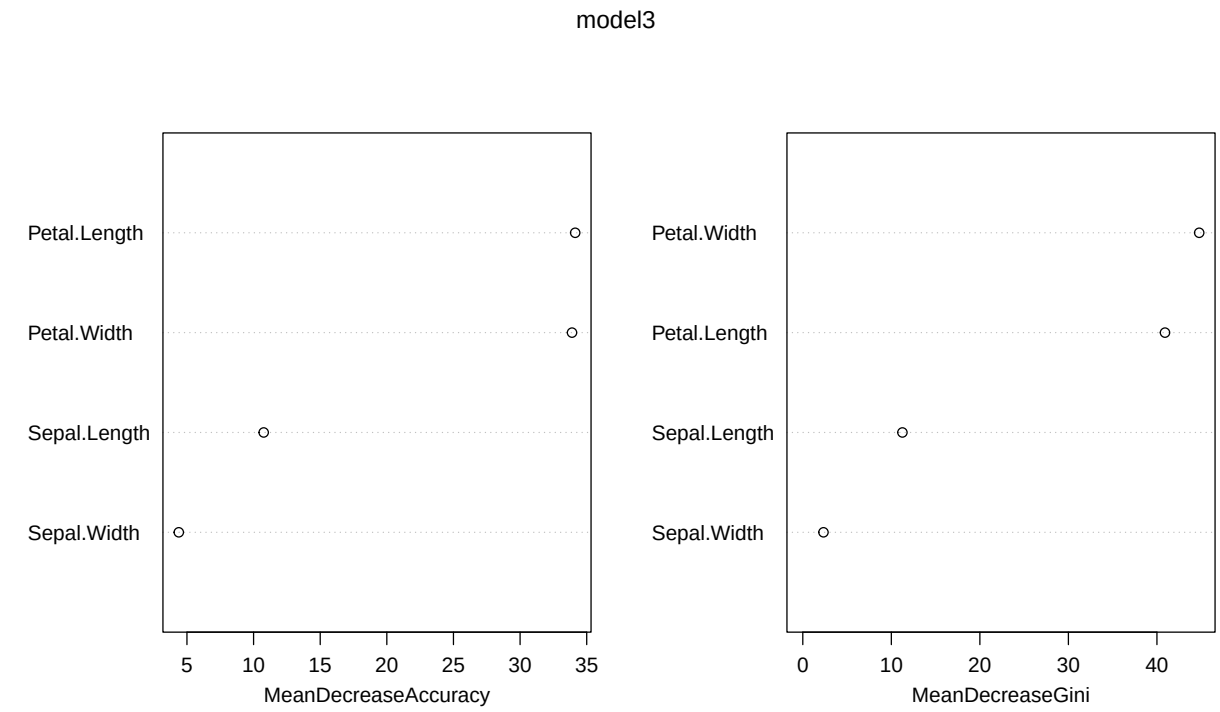
단계 2: 중요 변수 보기

```
importance(model3)
```

```
##                setosa versicolor virginica MeanDecreaseAccuracy
## Sepal.Length  6.874458  7.9146678  8.082254            10.759135
## Sepal.Width   4.642982  0.6861241  4.485976             4.387813
## Petal.Length 21.917953 33.7671161 27.496945            34.135733
## Petal.Width  22.641889 32.2664764 30.870675            33.897678
##              MeanDecreaseGini
## Sepal.Length         11.25043
## Sepal.Width           2.33405
## Petal.Length         40.91030
## Petal.Width          44.77354
```

단계 3: 중요 변수 시각화

```
varImpPlot(model3)
```



엔트포리(Entropy): 불확실성

```r
x1 <- 0.5; x2 <- 0.5
e1 <- -x1 * log2(x1) - x2 * log2(x2)
e1
```

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

```r
x1 <- 0.7; x2 <- 0.3
e2 <- -x1 * log2(x1) - x2 * log2(x2)
e2
```

```
## [1] 0.8812909
```

최적의 파라미터(ntree, mtry) 찾기

단계 1: 속성값 생성

```r
ntree <- c(400, 500, 600)
mtry <- c(2:4)
param <- data.frame(n = ntree, m = mtry)
```

단계 2: 이중 for() 함수를 이용하여 모델 생성

```r
for(i in param$n) {
  cat('ntree =', i, '\n')
  for(j in param$m) {
    cat('mtry =', j, '\n')
    model_iris <- randomForest(Species ~ ., data = iris, ntree = i, mtry = j, na.action = na.omit)
  }
}
```

```
## ntree = 400
## mtry = 2
## mtry = 3
## mtry = 4
## ntree = 500
## mtry = 2
## mtry = 3
## mtry = 4
## ntree = 600
## mtry = 2
## mtry = 3
## mtry = 4
```

다항 분류 xgboost 모델 생성

단계 1: 패키지 설치

```r
# install.packages("xgboost")
library(xgboost)
```

단계 2: y 변수 생성

```
iris_label <- ifelse(iris$Species == 'setosa', 0,
                     ifelse(iris$Species == 'versicolor', 1, 2))
table(iris_label)
```

```
## iris_label
##  0  1  2
## 50 50 50
```

```
iris$label <- iris_label
```

단계 3: 데이터 셋 생성

```
idx <- sample(nrow(iris), 0.7 * nrow(iris))
train <- iris[idx, ]
test <- iris[-idx, ]
```

단계 4: matrix 객체 변환

```
train_mat <- as.matrix(train[-c(5:6)])
dim(train_mat)
```

```
## [1] 105   4
```

```
train_lab <- train$label
length(train_lab)
```

```
## [1] 105
```

단계 5: xgb.DMatrix 객체 변환

```
dtrain <- xgb.DMatrix(data = train_mat, label = train_lab)
```

단계 6: model 생성 - xgboost matrix 객체 이용

```
xgb_model <- xgboost(data = dtrain, max_depth = 2, eta = 1,
                     nthread = 2, nrounds = 2,
                     objective = "multi:softmax",
                     num_class = 3,
                     verbose = 0)
xgb_model
```

```
## ##### xgb.Booster
## raw: 8.8 Kb
## call:
##   xgb.train(params = params, data = dtrain, nrounds = nrounds,
##     watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
##     early_stopping_rounds = early_stopping_rounds, maximize = maximize,
##     save_period = save_period, save_name = save_name, xgb_model = xgb_model,
##     callbacks = callbacks, max_depth = 2, eta = 1, nthread = 2,
##     objective = "multi:softmax", num_class = 3)
## params (as set within xgb.train):
##   max_depth = "2", eta = "1", nthread = "2", objective = "multi:softmax", num_class = "3", validate_
```

```
## xgb.attributes:
##   niter
## callbacks:
##   cb.evaluation.log()
## # of features: 4
## niter: 2
## nfeatures : 4
## evaluation_log:
##  iter train_mlogloss
##     1      0.2752219
##     2      0.1308610
```

단계 7: testset 생성

```
test_mat <- as.matrix(test[-c(5:6)])
dim(test_mat)
```

```
## [1] 45  4
```

```
test_lab <- test$label
length(test_lab)
```

```
## [1] 45
```

단계 8: model prediction

```
pred_iris <- predict(xgb_model, test_mat)
```

단계 9: confusion matrix

```
table(pred_iris, test_lab)
```

```
##          test_lab
## pred_iris  0  1  2
##         0 12  0  0
##         1  0 15  0
##         2  0  2 16
```

단계 10: 모델 성능평가1 - Accuracy

```
(19 + 13 + 12) / length(test_lab)
```

```
## [1] 0.9777778
```

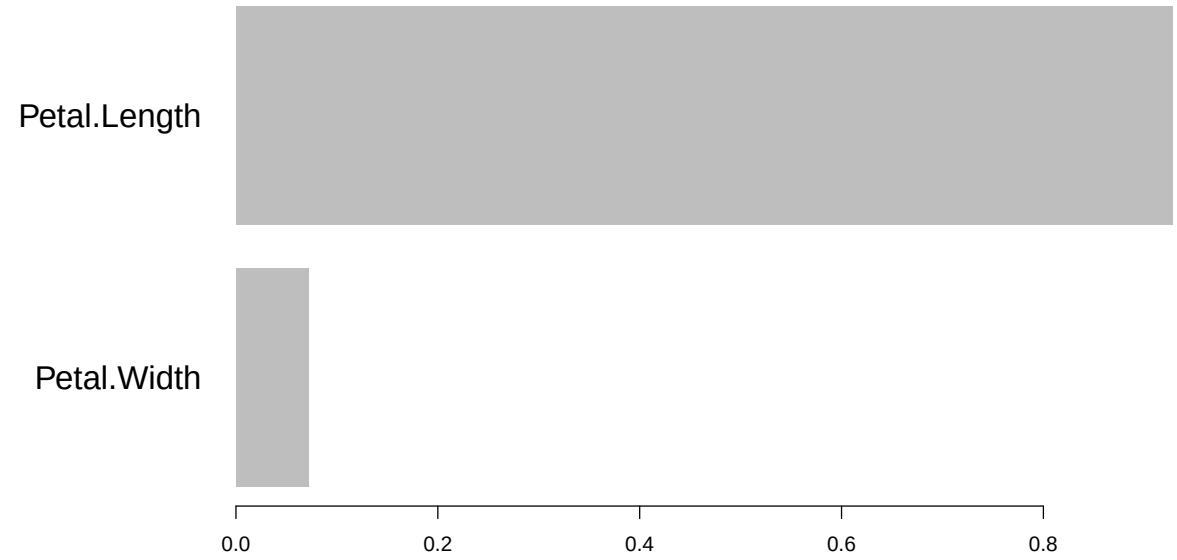
단계 11: model의 중요 변수(feature)와 영향력 보기

```
importance_matrix <- xgb.importance(colnames(train_mat),
                                    model = xgb_model)
importance_matrix
```

```
##         Feature       Gain     Cover Frequency
## 1: Petal.Length 0.92801485 0.7218847 0.6363636
## 2:  Petal.Width 0.07198515 0.2781153 0.3636364
```

단계 12: 중요 변수 시각화

```
xgb.plot.importance(importance_matrix)
```



간단한 인공신경망 모델 생성

단계 1: 패키지 설치

```
# install.packages("nnet")
library(nnet)
```

단계 2: 데이터 셋 생성

```
df = data.frame(    # 데이터프레임 생성 - 입력 변수(x)와 출력변수(y)
  x2 = c(1:6),
  x1 = c(6:1),
  y = factor(c('no', 'no', 'no', 'yes', 'yes', 'yes'))
)
str(df)
```

```
## 'data.frame':    6 obs. of  3 variables:
##  $ x2: int  1 2 3 4 5 6
##  $ x1: int  6 5 4 3 2 1
##  $ y : Factor w/ 2 levels "no","yes": 1 1 1 2 2 2
```

단계 3: 인공신경망 모델 생성

```
model_net = nnet(y ~ ., df, size = 1)
```

```
## # weights:  5
## initial  value 4.226573
## iter  10 value 4.154481
## iter  20 value 0.203251
## iter  30 value 0.003129
## iter  40 value 0.001305
## iter  50 value 0.000472
## iter  60 value 0.000317
## iter  70 value 0.000276
## iter  80 value 0.000152
## iter  90 value 0.000102
## final  value 0.000067
## converged
```

단계 4: 모델 결과 변수 보기

```
model_net
```

```
## a 2-1-1 network with 5 weights
## inputs: x2 x1
## output(s): y
## options were - entropy fitting
```

단계 5: 가중치(weights)보기

```
summary(model_net)
```

```
## a 2-1-1 network with 5 weights
## options were - entropy fitting
##  b->h1 i1->h1 i2->h1
##  -0.38  11.70 -10.90
##   b->o  h1->o
## -11.40  22.81
```

단계 6: 분류모델의 적합값 보기

```
model_net$fitted.values
```

```
##           [,1]
## 1 1.115227e-05
## 2 1.115227e-05
## 3 1.118739e-05
## 4 9.999889e-01
## 5 9.999889e-01
## 6 9.999889e-01
```

단계 7: 분류모델의 예측치 생성과 분류 정확도

```
p <- predict(model_net, df, type = "class")
table(p, df$y)
```

```
##
## p     no yes
##   no   3   0
##   yes  0   3
```

iris 데이터 셋을이용한 인공신경망 모델 생성

단계 1: 데이터 셋 생성

```
data(iris)
idx = sample(1:nrow(iris), 0.7 * nrow(iris))
training = iris[idx, ]
testing = iris[-idx, ]
nrow(training)
```

```
## [1] 105
```

```
nrow(testing)
```

```
## [1] 45
```

단계 2: 인공신경망 모델(은닉층 1개와 은닉층 3개) 생성

```
model_net_iris1 = nnet(Species ~ ., training, size = 1)
```

```
## # weights:  11
## initial  value 118.257386
## iter  10 value 51.335769
## iter  20 value 51.293299
## final  value 51.292987
## converged
```

```
model_net_iris1
```

```
## a 4-1-3 network with 11 weights
## inputs: Sepal.Length Sepal.Width Petal.Length Petal.Width
## output(s): Species
## options were - softmax modelling
```

```
model_net_iris3 = nnet(Species ~ ., training, size = 3)
```

```
## # weights:  27
## initial  value 123.023291
## iter  10 value 51.433892
## iter  20 value 51.293079
## final  value 51.292892
## converged
```

```
model_net_iris3
```

```
## a 4-3-3 network with 27 weights
## inputs: Sepal.Length Sepal.Width Petal.Length Petal.Width
## output(s): Species
## options were - softmax modelling
```

단계 3: 가중치 네트워크 보기 - 은닉층 1개 신경망 모델

```
summary(model_net_iris1)
```

```
## a 4-1-3 network with 11 weights
## options were - softmax modelling
##  b->h1 i1->h1 i2->h1 i3->h1 i4->h1
##  -1.88  -2.56  -7.55  16.54   9.25
##  b->o1 h1->o1
##  20.49 -28.61
##  b->o2 h1->o2
## -10.71  15.46
##  b->o3 h1->o3
##  -9.03  13.78
```

단계 4:가중치 네트워크 보기 - 은닉층 3개 신경망 모델

```
summary(model_net_iris3)
```

```
## a 4-3-3 network with 27 weights
## options were - softmax modelling
##  b->h1 i1->h1 i2->h1 i3->h1 i4->h1
##  -5.72 -28.75 -23.87  -0.36   1.56
##  b->h2 i1->h2 i2->h2 i3->h2 i4->h2
##   5.71   8.02  31.08 -51.21 -24.45
##  b->h3 i1->h3 i2->h3 i3->h3 i4->h3
##  -6.15 -17.93  -4.17   3.55  11.34
##  b->o1 h1->o1 h2->o1 h3->o1
## -16.64  12.33  34.81 -53.75
##  b->o2 h1->o2 h2->o2 h3->o2
##   8.53  -4.36 -24.42   6.34
##  b->o3 h1->o3 h2->o3 h3->o3
##   8.53  -7.66  -8.74  48.12
```

단계 5: 분류모델 평가

```
table(predict(model_net_iris1, testing, type = "class"), testing$Species)
```

```
##
##             setosa versicolor virginica
##   setosa        19          0         0
##   virginica      0         13        13
```

```
table(predict(model_net_iris3, testing, type = "class"), testing$Species)
```

```
##
##              setosa versicolor virginica
##   setosa         19          0         0
##   versicolor      0          7         8
##   virginica       0          6         5
```

neuralnet 패키지를 이용한 인공신경망 모델 생성

단계 1: 패키지 설치

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

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

```
## The following object is masked from 'package:ROCR':
##
##     prediction
```

단계 2: 데이터 셋 생성

```
data("iris")
idx = sample(1:nrow(iris), 0.7 * nrow(iris))
training_iris = iris[idx, ]
testing_iris = iris[-idx, ]
dim(training_iris)
```

```
## [1] 105   5
```

```
dim(testing_iris)
```

```
## [1] 45  5
```

단계 3: 수치형으로 칼럼 생성

```
training_iris$Species2[training_iris$Species == 'setosa'] <- 1
training_iris$Species2[training_iris$Species == 'versicolor'] <- 2
training_iris$Species2[training_iris$Species == 'virginica'] <- 3
training_iris$Species <- NULL
head(training_iris)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width Species2
## 51           7.0         3.2          4.7         1.4        2
## 149          6.2         3.4          5.4         2.3        3
## 11           5.4         3.7          1.5         0.2        1
## 69           6.2         2.2          4.5         1.5        2
## 81           5.5         2.4          3.8         1.1        2
## 124          6.3         2.7          4.9         1.8        3
```

```
testing_iris$Species2[testing_iris$Species == 'setosa'] <- 1
testing_iris$Species2[testing_iris$Species == 'versicolor'] <- 2
testing_iris$Species2[testing_iris$Species == 'virginica'] <- 3
testing_iris$Species <- NULL
head(testing_iris)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species2
## 1           5.1         3.5          1.4         0.2        1
## 4           4.6         3.1          1.5         0.2        1
## 8           5.0         3.4          1.5         0.2        1
## 10          4.9         3.1          1.5         0.1        1
## 15          5.8         4.0          1.2         0.2        1
```

```
## 18          5.1         3.5          1.4         0.3        1
```

단계 4: 데이터 정규화

단계 4-1: 정규화 함수 정의

```
normal <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
```

단계 4-2: 정규화 함수를 이용하여 학습데이터와/검정데이터 정규화

```
training_nor <- as.data.frame(lapply(training_iris, normal))
summary(training_nor)
```

```
##   Sepal.Length     Sepal.Width      Petal.Length      Petal.Width
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.2222   1st Qu.:0.2727   1st Qu.:0.07143   1st Qu.:0.08333
##  Median :0.4167   Median :0.3636   Median :0.55357   Median :0.50000
##  Mean   :0.4228   Mean   :0.3918   Mean   :0.45782   Mean   :0.45317
##  3rd Qu.:0.5833   3rd Qu.:0.5000   3rd Qu.:0.71429   3rd Qu.:0.70833
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
##     Species2
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.500
##  Mean   :0.481
##  3rd Qu.:1.000
##  Max.   :1.000
```

```
testing_nor <- as.data.frame(lapply(testing_iris, normal))
summary(testing_nor)
```

```
##   Sepal.Length     Sepal.Width      Petal.Length     Petal.Width
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.2258   1st Qu.:0.4000   1st Qu.:0.1186   1st Qu.:0.09091
##  Median :0.3871   Median :0.5000   Median :0.6102   Median :0.59091
##  Mean   :0.4172   Mean   :0.5233   Mean   :0.5047   Mean   :0.51212
##  3rd Qu.:0.5484   3rd Qu.:0.6500   3rd Qu.:0.7627   3rd Qu.:0.77273
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##     Species2
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.5000
##  Mean   :0.5444
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

단계 5: 인공신경망 모델 생성 - 은닉 노드 1개

```
model_net = neuralnet(Species2 ~ Sepal.Length + Sepal.Width +
                      Petal.Length + Petal.Width,
                      data = training_nor, hidden = 1)
plot(model_net)
```

단계 6: 분류모델 성능 평가

단계 6-1: 모델의 예측치 생성 - compute() 함수 이용

```
model_result <- compute(model_net, testing_nor[c(1:4)])
```

단계 6-2: 상관관계 분석 - 상관계수로 두 변수 간 선형관계의 강도 측정

```
cor(model_result$net.result, testing_nor$Species2)
```

```
##           [,1]
## [1,] 0.962226
```

단계 7: 분류모델 성능 향상 - 은닉층 노드 2개 지정, backprop 속성 적용

단계 7-1: 인공신경망 모델 생성

```
model_net2 = neuralnet(Species2 ~ Sepal.Length + Sepal.Width +
                         Petal.Length + Petal.Width,
                       data = training_nor, hidden = 2,
                       algorithm = "backprop", learningrate = 0.01)
```

단계 7-2: 분류모델 예측치 생성과 평가

```
model_result <- compute(model_net, testing_nor[c(1:4)])
cor(model_result$net.result, testing_nor$Species2)
```

```
##           [,1]
## [1,] 0.962226
```