04.decision\_tree\_initail\_model

Youngjin\_Lee

2022 3 10

Table of Contents

##### <https://www.tidymodels.org/start/models/>

# 1.초기 모델(Initia Model)

## 유니버설 은행 사례 Figure 9.9

tidyverse: ggplot2, purrr, tibble 3.0.3,  
dplyr, tidyr, stringr, readr, forcats  
—  
tidymodels: broom, recipes, dials, rsample, infer,  
tune, modeldata, workflows, parsnip,  
yardstick  
—

## 01.데이터 불러오기

textbook p.5 ~ 12 참조

bank\_tb <- read\_csv('../data/UniversalBank.csv',   
 col\_names = TRUE,  
 locale=locale('ko', encoding='euc-kr'),  
 na=".") %>% # csv 데이터 읽어오기  
 mutate\_if(is.character, as.factor)

## Rows: 5000 Columns: 14  
## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## dbl (14): ID, Age, Experience, Income, ZIP Code, Family, CCAvg, Education, M...  
##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

str(bank\_tb)

## spec\_tbl\_df [5,000 x 14] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ ID : num [1:5000] 1 2 3 4 5 6 7 8 9 10 ...  
## $ Age : num [1:5000] 25 45 39 35 35 37 53 50 35 34 ...  
## $ Experience : num [1:5000] 1 19 15 9 8 13 27 24 10 9 ...  
## $ Income : num [1:5000] 49 34 11 100 45 29 72 22 81 180 ...  
## $ ZIP Code : num [1:5000] 91107 90089 94720 94112 91330 ...  
## $ Family : num [1:5000] 4 3 1 1 4 4 2 1 3 1 ...  
## $ CCAvg : num [1:5000] 1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...  
## $ Education : num [1:5000] 1 1 1 2 2 2 2 3 2 3 ...  
## $ Mortgage : num [1:5000] 0 0 0 0 0 155 0 0 104 0 ...  
## $ Personal Loan : num [1:5000] 0 0 0 0 0 0 0 0 0 1 ...  
## $ Securities Account: num [1:5000] 1 1 0 0 0 0 0 0 0 0 ...  
## $ CD Account : num [1:5000] 0 0 0 0 0 0 0 0 0 0 ...  
## $ Online : num [1:5000] 0 0 0 0 0 1 1 0 1 0 ...  
## $ CreditCard : num [1:5000] 0 0 0 0 1 0 0 1 0 0 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. ID = col\_double(),  
## .. Age = col\_double(),  
## .. Experience = col\_double(),  
## .. Income = col\_double(),  
## .. `ZIP Code` = col\_double(),  
## .. Family = col\_double(),  
## .. CCAvg = col\_double(),  
## .. Education = col\_double(),  
## .. Mortgage = col\_double(),  
## .. `Personal Loan` = col\_double(),  
## .. `Securities Account` = col\_double(),  
## .. `CD Account` = col\_double(),  
## .. Online = col\_double(),  
## .. CreditCard = col\_double()  
## .. )  
## - attr(\*, "problems")=<externalptr>

head(bank\_tb)

## # A tibble: 6 x 14  
## ID Age Experience Income `ZIP Code` Family CCAvg Education Mortgage  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 25 1 49 91107 4 1.6 1 0  
## 2 2 45 19 34 90089 3 1.5 1 0  
## 3 3 39 15 11 94720 1 1 1 0  
## 4 4 35 9 100 94112 1 2.7 2 0  
## 5 5 35 8 45 91330 4 1 2 0  
## 6 6 37 13 29 92121 4 0.4 2 155  
## # ... with 5 more variables: `Personal Loan` <dbl>, `Securities Account` <dbl>,  
## # `CD Account` <dbl>, Online <dbl>, CreditCard <dbl>

## 02.data 전처리

변수명 수정 (공란이 있을 경우에 변수명 수정)

bank\_tb <- bank\_tb %>%  
 rename(c('Personal\_Loan'= 'Personal Loan',  
 'CD\_Account' = 'CD Account',  
 'Securities\_Account' = 'Securities Account'))  
str(bank\_tb)

## spec\_tbl\_df [5,000 x 14] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ ID : num [1:5000] 1 2 3 4 5 6 7 8 9 10 ...  
## $ Age : num [1:5000] 25 45 39 35 35 37 53 50 35 34 ...  
## $ Experience : num [1:5000] 1 19 15 9 8 13 27 24 10 9 ...  
## $ Income : num [1:5000] 49 34 11 100 45 29 72 22 81 180 ...  
## $ ZIP Code : num [1:5000] 91107 90089 94720 94112 91330 ...  
## $ Family : num [1:5000] 4 3 1 1 4 4 2 1 3 1 ...  
## $ CCAvg : num [1:5000] 1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...  
## $ Education : num [1:5000] 1 1 1 2 2 2 2 3 2 3 ...  
## $ Mortgage : num [1:5000] 0 0 0 0 0 155 0 0 104 0 ...  
## $ Personal\_Loan : num [1:5000] 0 0 0 0 0 0 0 0 0 1 ...  
## $ Securities\_Account: num [1:5000] 1 1 0 0 0 0 0 0 0 0 ...  
## $ CD\_Account : num [1:5000] 0 0 0 0 0 0 0 0 0 0 ...  
## $ Online : num [1:5000] 0 0 0 0 0 1 1 0 1 0 ...  
## $ CreditCard : num [1:5000] 0 0 0 0 1 0 0 1 0 0 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. ID = col\_double(),  
## .. Age = col\_double(),  
## .. Experience = col\_double(),  
## .. Income = col\_double(),  
## .. `ZIP Code` = col\_double(),  
## .. Family = col\_double(),  
## .. CCAvg = col\_double(),  
## .. Education = col\_double(),  
## .. Mortgage = col\_double(),  
## .. `Personal Loan` = col\_double(),  
## .. `Securities Account` = col\_double(),  
## .. `CD Account` = col\_double(),  
## .. Online = col\_double(),  
## .. CreditCard = col\_double()  
## .. )  
## - attr(\*, "problems")=<externalptr>

head(bank\_tb)

## # A tibble: 6 x 14  
## ID Age Experience Income `ZIP Code` Family CCAvg Education Mortgage  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 25 1 49 91107 4 1.6 1 0  
## 2 2 45 19 34 90089 3 1.5 1 0  
## 3 3 39 15 11 94720 1 1 1 0  
## 4 4 35 9 100 94112 1 2.7 2 0  
## 5 5 35 8 45 91330 4 1 2 0  
## 6 6 37 13 29 92121 4 0.4 2 155  
## # ... with 5 more variables: Personal\_Loan <dbl>, Securities\_Account <dbl>,  
## # CD\_Account <dbl>, Online <dbl>, CreditCard <dbl>

범주형 변수(factor)로 인식하게 변환 결과변수(class)에서 관심있는 변수를 1번으로 세팅

bank\_tb <- bank\_tb %>%  
 mutate(Personal\_Loan = factor(Personal\_Loan,   
 levels = c(1, 0), #관심변수=Yes   
 labels = c("Yes", "No"))) %>%  
 mutate(Securities\_Account = factor(Securities\_Account,   
 levels = c(0,1),  
 labels = c("No", "Yes"))) %>%  
 mutate(CD\_Account = factor(CD\_Account,   
 levels = c(0,1),  
 labels = c("No", "Yes"))) %>%  
 mutate(Online = factor(Online,  
 levels = c(0,1),  
 labels = c("No", "Yes"))) %>%  
 mutate(CreditCard = factor(CreditCard,  
 levels = c(0,1),  
 labels = c("No", "Yes"))) %>%  
 mutate(Education = factor(Education ,  
 levels = c(1:3),  
 labels = c("Undergrad",   
 "Graduate",   
 "Professional")))  
  
str(bank\_tb)

## spec\_tbl\_df [5,000 x 14] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ ID : num [1:5000] 1 2 3 4 5 6 7 8 9 10 ...  
## $ Age : num [1:5000] 25 45 39 35 35 37 53 50 35 34 ...  
## $ Experience : num [1:5000] 1 19 15 9 8 13 27 24 10 9 ...  
## $ Income : num [1:5000] 49 34 11 100 45 29 72 22 81 180 ...  
## $ ZIP Code : num [1:5000] 91107 90089 94720 94112 91330 ...  
## $ Family : num [1:5000] 4 3 1 1 4 4 2 1 3 1 ...  
## $ CCAvg : num [1:5000] 1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...  
## $ Education : Factor w/ 3 levels "Undergrad","Graduate",..: 1 1 1 2 2 2 2 3 2 3 ...  
## $ Mortgage : num [1:5000] 0 0 0 0 0 155 0 0 104 0 ...  
## $ Personal\_Loan : Factor w/ 2 levels "Yes","No": 2 2 2 2 2 2 2 2 2 1 ...  
## $ Securities\_Account: Factor w/ 2 levels "No","Yes": 2 2 1 1 1 1 1 1 1 1 ...  
## $ CD\_Account : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Online : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 2 1 2 1 ...  
## $ CreditCard : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 1 2 1 1 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. ID = col\_double(),  
## .. Age = col\_double(),  
## .. Experience = col\_double(),  
## .. Income = col\_double(),  
## .. `ZIP Code` = col\_double(),  
## .. Family = col\_double(),  
## .. CCAvg = col\_double(),  
## .. Education = col\_double(),  
## .. Mortgage = col\_double(),  
## .. `Personal Loan` = col\_double(),  
## .. `Securities Account` = col\_double(),  
## .. `CD Account` = col\_double(),  
## .. Online = col\_double(),  
## .. CreditCard = col\_double()  
## .. )  
## - attr(\*, "problems")=<externalptr>

head(bank\_tb)

## # A tibble: 6 x 14  
## ID Age Experience Income `ZIP Code` Family CCAvg Education Mortgage  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct> <dbl>  
## 1 1 25 1 49 91107 4 1.6 Undergrad 0  
## 2 2 45 19 34 90089 3 1.5 Undergrad 0  
## 3 3 39 15 11 94720 1 1 Undergrad 0  
## 4 4 35 9 100 94112 1 2.7 Graduate 0  
## 5 5 35 8 45 91330 4 1 Graduate 0  
## 6 6 37 13 29 92121 4 0.4 Graduate 155  
## # ... with 5 more variables: Personal\_Loan <fct>, Securities\_Account <fct>,  
## # CD\_Account <fct>, Online <fct>, CreditCard <fct>

필요없는 변수제거: ID, 우편번호 제거 recipe에서 제거할 수도 있음

bank\_tb <- bank\_tb %>%  
 select(-c(ID, `ZIP Code`))   
  
str(bank\_tb)

## tibble [5,000 x 12] (S3: tbl\_df/tbl/data.frame)  
## $ Age : num [1:5000] 25 45 39 35 35 37 53 50 35 34 ...  
## $ Experience : num [1:5000] 1 19 15 9 8 13 27 24 10 9 ...  
## $ Income : num [1:5000] 49 34 11 100 45 29 72 22 81 180 ...  
## $ Family : num [1:5000] 4 3 1 1 4 4 2 1 3 1 ...  
## $ CCAvg : num [1:5000] 1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...  
## $ Education : Factor w/ 3 levels "Undergrad","Graduate",..: 1 1 1 2 2 2 2 3 2 3 ...  
## $ Mortgage : num [1:5000] 0 0 0 0 0 155 0 0 104 0 ...  
## $ Personal\_Loan : Factor w/ 2 levels "Yes","No": 2 2 2 2 2 2 2 2 2 1 ...  
## $ Securities\_Account: Factor w/ 2 levels "No","Yes": 2 2 1 1 1 1 1 1 1 1 ...  
## $ CD\_Account : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Online : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 2 1 2 1 ...  
## $ CreditCard : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 1 2 1 1 ...

head(bank\_tb)

## # A tibble: 6 x 12  
## Age Experience Income Family CCAvg Education Mortgage Personal\_Loan  
## <dbl> <dbl> <dbl> <dbl> <dbl> <fct> <dbl> <fct>   
## 1 25 1 49 4 1.6 Undergrad 0 No   
## 2 45 19 34 3 1.5 Undergrad 0 No   
## 3 39 15 11 1 1 Undergrad 0 No   
## 4 35 9 100 1 2.7 Graduate 0 No   
## 5 35 8 45 4 1 Graduate 0 No   
## 6 37 13 29 4 0.4 Graduate 155 No   
## # ... with 4 more variables: Securities\_Account <fct>, CD\_Account <fct>,  
## # Online <fct>, CreditCard <fct>

## 03.데이터 탐색(EDA)

데이터 탐색: 범주형, 연속형 구분 skimr::skim() - package명을 앞에 써서 구분 패키지를 여러개 사용할 경우에 이름이 같은 경우도 있어서 구분이 필요할 경우에 [패키지명::]을 사용

bank\_tb %>%  
 skimr::skim()

Data summary

|  |  |
| --- | --- |
| Name | Piped data |
| Number of rows | 5000 |
| Number of columns | 12 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 6 |
| numeric | 6 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| --- | --- | --- | --- | --- | --- |
| Education | 0 | 1 | FALSE | 3 | Und: 2096, Pro: 1501, Gra: 1403 |
| Personal\_Loan | 0 | 1 | FALSE | 2 | No: 4520, Yes: 480 |
| Securities\_Account | 0 | 1 | FALSE | 2 | No: 4478, Yes: 522 |
| CD\_Account | 0 | 1 | FALSE | 2 | No: 4698, Yes: 302 |
| Online | 0 | 1 | FALSE | 2 | Yes: 2984, No: 2016 |
| CreditCard | 0 | 1 | FALSE | 2 | No: 3530, Yes: 1470 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Age | 0 | 1 | 45.34 | 11.46 | 23 | 35.0 | 45.0 | 55.0 | 67 | ▅▇▇▇▆ |
| Experience | 0 | 1 | 20.10 | 11.47 | -3 | 10.0 | 20.0 | 30.0 | 43 | ▅▇▇▇▅ |
| Income | 0 | 1 | 73.77 | 46.03 | 8 | 39.0 | 64.0 | 98.0 | 224 | ▇▇▃▂▁ |
| Family | 0 | 1 | 2.40 | 1.15 | 1 | 1.0 | 2.0 | 3.0 | 4 | ▇▇▁▆▆ |
| CCAvg | 0 | 1 | 1.94 | 1.75 | 0 | 0.7 | 1.5 | 2.5 | 10 | ▇▃▁▁▁ |
| Mortgage | 0 | 1 | 56.50 | 101.71 | 0 | 0.0 | 0.0 | 101.0 | 635 | ▇▂▁▁▁ |

bank\_tb %>%  
 group\_by(Personal\_Loan) %>%  
 skimr::skim()

Data summary

|  |  |
| --- | --- |
| Name | Piped data |
| Number of rows | 5000 |
| Number of columns | 12 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 5 |
| numeric | 6 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | Personal\_Loan |

**Variable type: factor**

| skim\_variable | Personal\_Loan | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| --- | --- | --- | --- | --- | --- | --- |
| Education | Yes | 0 | 1 | FALSE | 3 | Pro: 205, Gra: 182, Und: 93 |
| Education | No | 0 | 1 | FALSE | 3 | Und: 2003, Pro: 1296, Gra: 1221 |
| Securities\_Account | Yes | 0 | 1 | FALSE | 2 | No: 420, Yes: 60 |
| Securities\_Account | No | 0 | 1 | FALSE | 2 | No: 4058, Yes: 462 |
| CD\_Account | Yes | 0 | 1 | FALSE | 2 | No: 340, Yes: 140 |
| CD\_Account | No | 0 | 1 | FALSE | 2 | No: 4358, Yes: 162 |
| Online | Yes | 0 | 1 | FALSE | 2 | Yes: 291, No: 189 |
| Online | No | 0 | 1 | FALSE | 2 | Yes: 2693, No: 1827 |
| CreditCard | Yes | 0 | 1 | FALSE | 2 | No: 337, Yes: 143 |
| CreditCard | No | 0 | 1 | FALSE | 2 | No: 3193, Yes: 1327 |

**Variable type: numeric**

| skim\_variable | Personal\_Loan | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Age | Yes | 0 | 1 | 45.07 | 11.59 | 26 | 35.0 | 45.0 | 55.00 | 65.0 | ▇▇▇▇▇ |
| Age | No | 0 | 1 | 45.37 | 11.45 | 23 | 35.0 | 45.0 | 55.00 | 67.0 | ▅▇▇▇▅ |
| Experience | Yes | 0 | 1 | 19.84 | 11.58 | 0 | 9.0 | 20.0 | 30.00 | 41.0 | ▇▇▇▇▆ |
| Experience | No | 0 | 1 | 20.13 | 11.46 | -3 | 10.0 | 20.0 | 30.00 | 43.0 | ▅▇▇▇▅ |
| Income | Yes | 0 | 1 | 144.75 | 31.58 | 60 | 122.0 | 142.5 | 172.00 | 203.0 | ▁▅▇▆▆ |
| Income | No | 0 | 1 | 66.24 | 40.58 | 8 | 35.0 | 59.0 | 84.00 | 224.0 | ▇▇▂▁▁ |
| Family | Yes | 0 | 1 | 2.61 | 1.12 | 1 | 2.0 | 3.0 | 4.00 | 4.0 | ▆▆▁▇▇ |
| Family | No | 0 | 1 | 2.37 | 1.15 | 1 | 1.0 | 2.0 | 3.00 | 4.0 | ▇▇▁▅▆ |
| CCAvg | Yes | 0 | 1 | 3.91 | 2.10 | 0 | 2.6 | 3.8 | 5.35 | 10.0 | ▅▇▇▃▁ |
| CCAvg | No | 0 | 1 | 1.73 | 1.57 | 0 | 0.6 | 1.4 | 2.30 | 8.8 | ▇▅▁▁▁ |
| Mortgage | Yes | 0 | 1 | 100.85 | 160.85 | 0 | 0.0 | 0.0 | 192.50 | 617.0 | ▇▁▁▁▁ |
| Mortgage | No | 0 | 1 | 51.79 | 92.04 | 0 | 0.0 | 0.0 | 98.00 | 635.0 | ▇▂▁▁▁ |

base accuracy yes 기준으로 0.096(yes인확률)

bank\_tb %>%   
 count(Personal\_Loan) %>%   
 mutate(prop = n/sum(n))

## # A tibble: 2 x 3  
## Personal\_Loan n prop  
## <fct> <int> <dbl>  
## 1 Yes 480 0.096  
## 2 No 4520 0.904

## 04.훈련용, 테스트용 데이터 분할: partition

데이터 partition

set.seed(123) # 시드 고정 (결과값을 유지)  
  
bank\_split <-   
 initial\_split(bank\_tb, prop=0.7, # 비율 기본값 7;3(prop=0.7), train\_data, test\_data  
 strata = Personal\_Loan) # 결과변수 비율반영, strata(층화표본) 기준  
   
bank\_split

## <Analysis/Assess/Total>  
## <3500/1500/5000>

training, test용 분리

train\_data <- training(bank\_split)  
test\_data <- testing(bank\_split)  
str(train\_data)

## tibble [3,500 x 12] (S3: tbl\_df/tbl/data.frame)  
## $ Age : num [1:3500] 25 45 35 53 34 65 29 48 59 60 ...  
## $ Experience : num [1:3500] 1 19 8 27 9 39 5 23 32 30 ...  
## $ Income : num [1:3500] 49 34 45 72 180 105 45 114 40 22 ...  
## $ Family : num [1:3500] 4 3 4 2 1 4 3 2 4 1 ...  
## $ CCAvg : num [1:3500] 1.6 1.5 1 1.5 8.9 2.4 0.1 3.8 2.5 1.5 ...  
## $ Education : Factor w/ 3 levels "Undergrad","Graduate",..: 1 1 2 2 3 3 2 3 2 3 ...  
## $ Mortgage : num [1:3500] 0 0 0 0 0 0 0 0 0 0 ...  
## $ Personal\_Loan : Factor w/ 2 levels "Yes","No": 2 2 2 2 1 2 2 2 2 2 ...  
## $ Securities\_Account: Factor w/ 2 levels "No","Yes": 2 2 1 1 1 1 1 2 1 1 ...  
## $ CD\_Account : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Online : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 1 2 1 2 2 ...  
## $ CreditCard : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 1 1 1 1 2 ...

str(test\_data)

## tibble [1,500 x 12] (S3: tbl\_df/tbl/data.frame)  
## $ Age : num [1:1500] 39 35 37 50 35 67 29 44 59 53 ...  
## $ Experience : num [1:1500] 15 9 13 24 10 41 5 18 35 28 ...  
## $ Income : num [1:1500] 11 100 29 22 81 112 62 43 35 41 ...  
## $ Family : num [1:1500] 1 1 4 1 3 1 1 2 1 2 ...  
## $ CCAvg : num [1:1500] 1 2.7 0.4 0.3 0.6 2 1.2 0.7 1.2 0.6 ...  
## $ Education : Factor w/ 3 levels "Undergrad","Graduate",..: 1 2 2 3 2 1 1 1 3 3 ...  
## $ Mortgage : num [1:1500] 0 0 155 0 104 0 260 163 122 193 ...  
## $ Personal\_Loan : Factor w/ 2 levels "Yes","No": 2 2 2 2 2 2 2 2 2 2 ...  
## $ Securities\_Account: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 2 1 1 ...  
## $ CD\_Account : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Online : Factor w/ 2 levels "No","Yes": 1 1 2 1 2 1 2 1 2 1 ...  
## $ CreditCard : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 1 1 1 1 1 ...

## 05.Model 만들기

textbook p.31 ~ 36 설명 참조 ### Model 만들기 모델 인자(argument) 확인

args(decision\_tree)

## function (mode = "unknown", engine = "rpart", cost\_complexity = NULL,   
## tree\_depth = NULL, min\_n = NULL)   
## NULL

tree\_model <-   
 decision\_tree() %>% # function cost\_complexity = NULL, tree\_depth = NULL, min\_n = NULL  
 set\_engine("rpart") %>%   
 set\_mode("classification")

### recipe 만들기(데이터 튜닝)

step\_dummy(all\_nominal(), -all\_outcomes()) : one-hot-ecoding 적용, 기본 step\_log(Gr\_Liv\_Area, base = 10) : 로그함수로 변환 step\_other(Neighborhood, threshold = 0.01) : 값이 적은 항목을 기타로 변환 step\_upsample(Personal\_Loan) # 데이터 균형화 step\_zv(all\_predictors()) : 단일 고유 값 (예 : 모두 0) 변수 제거. 특히, penalty 사용하는 모델에서 중요(logistic, SVM 등) step\_normalize(all\_numeric()) : 데이터 정규화

tree\_recipe <-   
 recipe(Personal\_Loan ~ ., data = train\_data) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
   
summary(tree\_recipe)

## # A tibble: 12 x 4  
## variable type role source   
## <chr> <chr> <chr> <chr>   
## 1 Age numeric predictor original  
## 2 Experience numeric predictor original  
## 3 Income numeric predictor original  
## 4 Family numeric predictor original  
## 5 CCAvg numeric predictor original  
## 6 Education nominal predictor original  
## 7 Mortgage numeric predictor original  
## 8 Securities\_Account nominal predictor original  
## 9 CD\_Account nominal predictor original  
## 10 Online nominal predictor original  
## 11 CreditCard nominal predictor original  
## 12 Personal\_Loan nominal outcome original

## 06.workflow 만들기

textbook p.39 설명참조

tree\_workflow <-   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(tree\_recipe)  
  
tree\_workflow

## == Workflow ====================================================================  
## Preprocessor: Recipe  
## Model: decision\_tree()  
##   
## -- Preprocessor ----------------------------------------------------------------  
## 1 Recipe Step  
##   
## \* step\_dummy()  
##   
## -- Model -----------------------------------------------------------------------  
## Decision Tree Model Specification (classification)  
##   
## Computational engine: rpart

## 07.Model 훈련

훈련데이터로 모델 훈련하기

tree\_train\_fit <-   
 tree\_workflow %>%  
 fit(data = train\_data)

모델 훈련 결과 확인

tree\_train\_fit %>%  
 extract\_fit\_parsnip() # `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.

## parsnip model object  
##   
## Fit time: 31ms   
## n= 3500   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 3500 323 No (0.09228571 0.90771429)   
## 2) Income>=100.5 822 296 No (0.36009732 0.63990268)   
## 4) Family>=2.5 214 47 Yes (0.78037383 0.21962617)   
## 8) Income>=114.5 147 2 Yes (0.98639456 0.01360544) \*  
## 9) Income< 114.5 67 22 No (0.32835821 0.67164179)   
## 18) CCAvg>=2.75 20 6 Yes (0.70000000 0.30000000) \*  
## 19) CCAvg< 2.75 47 8 No (0.17021277 0.82978723) \*  
## 5) Family< 2.5 608 129 No (0.21217105 0.78782895)   
## 10) Education\_Professional>=0.5 94 19 Yes (0.79787234 0.20212766)   
## 20) Income>=114.5 67 1 Yes (0.98507463 0.01492537) \*  
## 21) Income< 114.5 27 9 No (0.33333333 0.66666667) \*  
## 11) Education\_Professional< 0.5 514 54 No (0.10505837 0.89494163)   
## 22) Education\_Graduate>=0.5 63 11 Yes (0.82539683 0.17460317)   
## 44) Income>=116.5 46 0 Yes (1.00000000 0.00000000) \*  
## 45) Income< 116.5 17 6 No (0.35294118 0.64705882) \*  
## 23) Education\_Graduate< 0.5 451 2 No (0.00443459 0.99556541) \*  
## 3) Income< 100.5 2678 27 No (0.01008215 0.98991785)   
## 6) CCAvg>=2.95 160 27 No (0.16875000 0.83125000)   
## 12) CD\_Account\_Yes>=0.5 9 1 Yes (0.88888889 0.11111111) \*  
## 13) CD\_Account\_Yes< 0.5 151 19 No (0.12582781 0.87417219) \*  
## 7) CCAvg< 2.95 2518 0 No (0.00000000 1.00000000) \*

## 08.훈련모델 검정

textbook p.44 ~ 50 설명 참조 ### 예측결과표 생성

tree\_train\_pred <-   
 predict(tree\_train\_fit,   
 train\_data,   
 type = "prob") %>%  
 bind\_cols(predict(tree\_train\_fit,   
 train\_data)) %>%   
 bind\_cols(train\_data %>%   
 select(Personal\_Loan)) %>%  
 print()

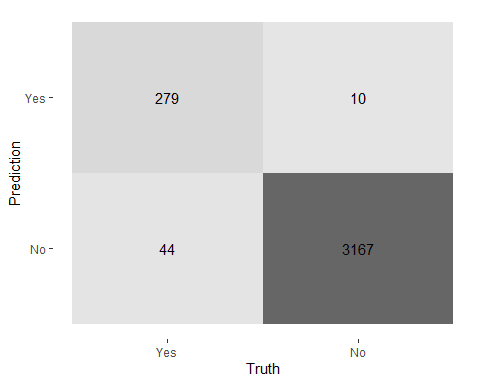
## # A tibble: 3,500 x 4  
## .pred\_Yes .pred\_No .pred\_class Personal\_Loan  
## <dbl> <dbl> <fct> <fct>   
## 1 0 1 No No   
## 2 0 1 No No   
## 3 0 1 No No   
## 4 0 1 No No   
## 5 0.985 0.0149 Yes Yes   
## 6 0.170 0.830 No No   
## 7 0 1 No No   
## 8 0.333 0.667 No No   
## 9 0 1 No No   
## 10 0 1 No No   
## # ... with 3,490 more rows

### 정오분류표(confusion matrix) 만들기

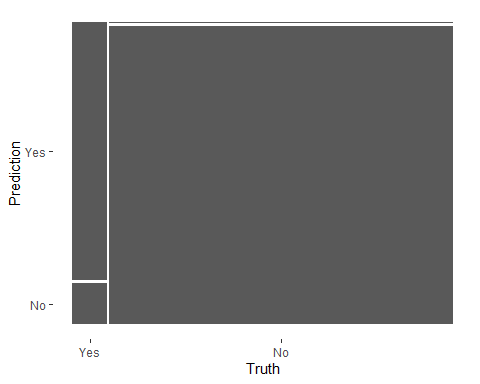
tree\_train\_conf <-  
 tree\_train\_pred %>%  
 conf\_mat(truth = Personal\_Loan,   
 estimate = .pred\_class)  
  
tree\_train\_conf

## Truth  
## Prediction Yes No  
## Yes 279 10  
## No 44 3167

autoplot(tree\_train\_conf, type = "heatmap") # mosaic



autoplot(tree\_train\_conf, type = "mosaic")



summary(tree\_train\_conf)

## # A tibble: 13 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 accuracy binary 0.985   
## 2 kap binary 0.903   
## 3 sens binary 0.864   
## 4 spec binary 0.997   
## 5 ppv binary 0.965   
## 6 npv binary 0.986   
## 7 mcc binary 0.905   
## 8 j\_index binary 0.861   
## 9 bal\_accuracy binary 0.930   
## 10 detection\_prevalence binary 0.0826  
## 11 precision binary 0.965   
## 12 recall binary 0.864   
## 13 f\_meas binary 0.912

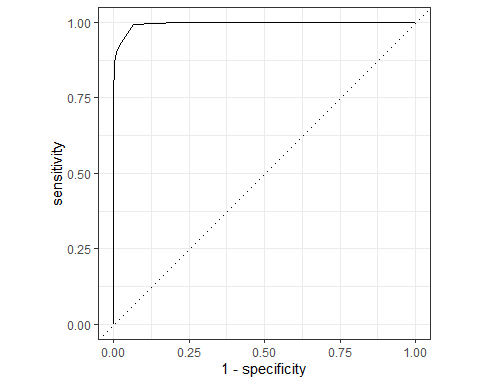
f1: 재현율(Recall)(↑)과 정밀도(Precision)(↑) 재현율(Recall): 실제 Class 중에 잘 맞춘 것(=TPR=민감도) 정밀도(Precision): 예측 Class 중에 잘 맞춘 것 정확도 (Accuracy) : 클래스 0과 1 모두를 정확하게 분류 ### ACU(area under the curve): ROC 정확도

tree\_train\_pred %>%  
 roc\_auc(truth = Personal\_Loan,   
 .pred\_Yes)

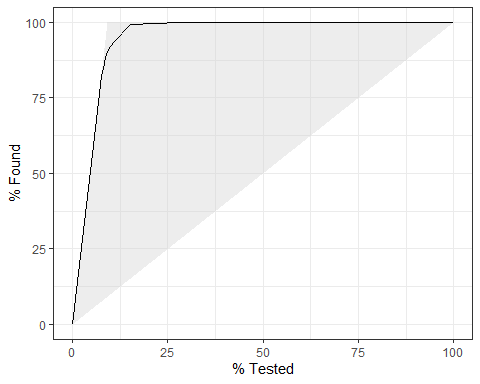
## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 roc\_auc binary 0.995

### ROC 커브

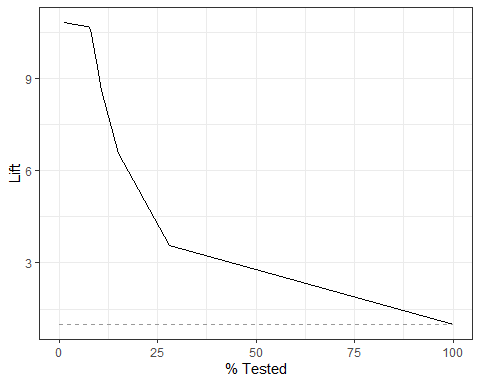
train\_auc <-  
 tree\_train\_pred %>%  
 roc\_curve(truth = Personal\_Loan,   
 estimate = .pred\_Yes) %>%   
 mutate(model = "train\_auc")  
  
autoplot(train\_auc)

 ### gain 커브

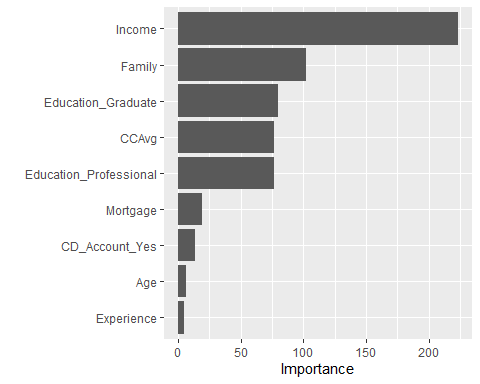
tree\_train\_pred %>%  
 gain\_curve(truth = Personal\_Loan,   
 estimate = .pred\_Yes) %>%  
 autoplot()

 ### lift 커브

tree\_train\_pred %>%  
 lift\_curve(truth = Personal\_Loan,   
 estimate = .pred\_Yes) %>%  
 autoplot()

 ### 중요변수 확인

tree\_train\_fit %>%   
 extract\_fit\_parsnip() %>% # pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
 vip()

 ## 09.테스트 데이터 검정 구축된 모델에 test data로 검정 last\_fit 사용 data: bank\_split 사용(테스트는 전체데이터 = train + test)

tree\_test\_fit <-   
 tree\_workflow %>%  
 last\_fit(bank\_split)   
  
tree\_test\_fit

## # Resampling results  
## # Manual resampling   
## # A tibble: 1 x 6  
## splits id .metrics .notes .predictions .workflow   
## <list> <chr> <list> <list> <list> <list>   
## 1 <split [3500/1500]> train/test split <tibble> <tibble> <tibble> <workflow>

### 예측결과 자동생성: collect\_predictions()

tree\_test\_pred <-   
 tree\_test\_fit %>%  
 collect\_predictions()  
  
tree\_test\_pred

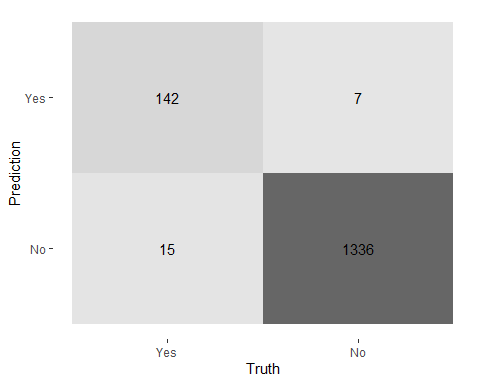
## # A tibble: 1,500 x 7  
## id .pred\_Yes .pred\_No .row .pred\_class Personal\_Loan .config   
## <chr> <dbl> <dbl> <int> <fct> <fct> <chr>   
## 1 train/test split 0 1 3 No No Preproce~  
## 2 train/test split 0 1 4 No No Preproce~  
## 3 train/test split 0 1 6 No No Preproce~  
## 4 train/test split 0 1 8 No No Preproce~  
## 5 train/test split 0 1 9 No No Preproce~  
## 6 train/test split 0.00443 0.996 15 No No Preproce~  
## 7 train/test split 0 1 23 No No Preproce~  
## 8 train/test split 0 1 24 No No Preproce~  
## 9 train/test split 0 1 31 No No Preproce~  
## 10 train/test split 0 1 33 No No Preproce~  
## # ... with 1,490 more rows

### 정오분류표(confusion matrix) 만들기

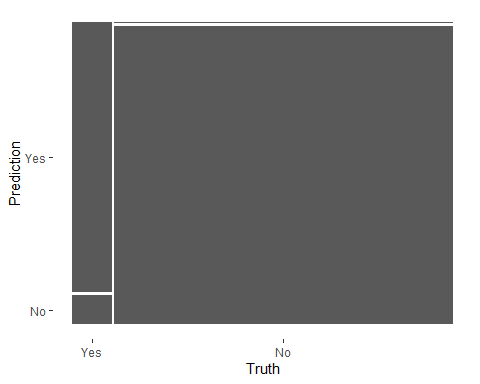
tree\_test\_conf <-  
 tree\_test\_pred %>%  
 conf\_mat(truth = Personal\_Loan,   
 estimate = .pred\_class)  
  
tree\_test\_conf

## Truth  
## Prediction Yes No  
## Yes 142 7  
## No 15 1336

autoplot(tree\_test\_conf, type = "heatmap") # mosaic



autoplot(tree\_test\_conf, type = "mosaic")



summary(tree\_test\_conf)

## # A tibble: 13 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 accuracy binary 0.985   
## 2 kap binary 0.920   
## 3 sens binary 0.904   
## 4 spec binary 0.995   
## 5 ppv binary 0.953   
## 6 npv binary 0.989   
## 7 mcc binary 0.920   
## 8 j\_index binary 0.899   
## 9 bal\_accuracy binary 0.950   
## 10 detection\_prevalence binary 0.0993  
## 11 precision binary 0.953   
## 12 recall binary 0.904   
## 13 f\_meas binary 0.928

f1: 재현율(Recall)(↑)과 정밀도(Precision)(↑) 재현율(Recall): 실제 Class 중에 잘 맞춘 것(=TPR=민감도) 정밀도(Precision): 예측 Class 중에 잘 맞춘 것 정확도 (Accuracy) : 클래스 0과 1 모두를 정확하게 분류

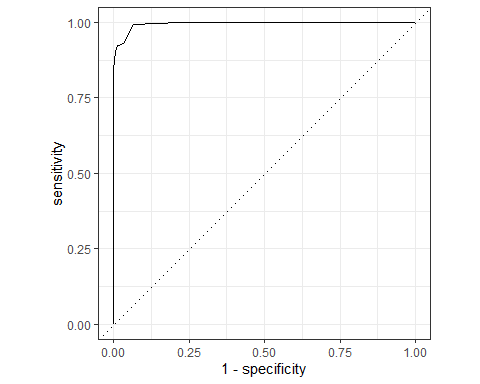
### ACU(area under the curve): ROC 정확도

tree\_test\_pred %>%  
 roc\_auc(truth = Personal\_Loan,   
 .pred\_Yes)

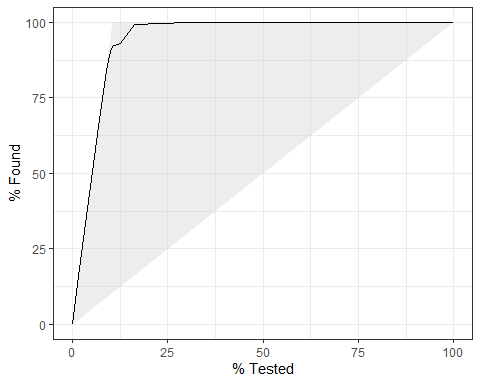
## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 roc\_auc binary 0.995

### ROC 커브

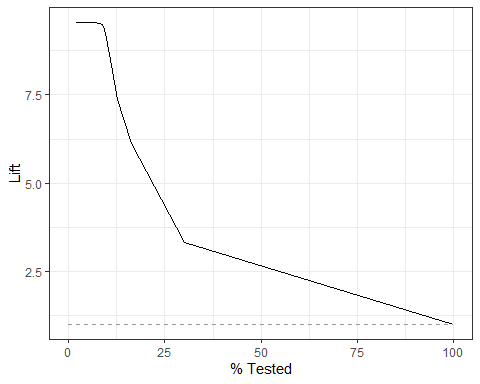
test\_auc <-  
 tree\_test\_pred %>%  
 roc\_curve(truth = Personal\_Loan,   
 estimate = .pred\_Yes) %>%   
 mutate(model = "test\_auc")  
  
autoplot(test\_auc)

 ### gain 커브

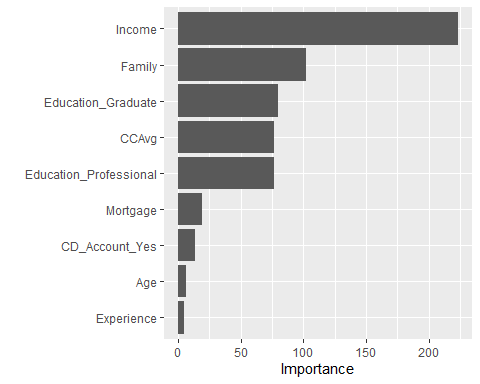
tree\_test\_pred %>%  
 gain\_curve(truth = Personal\_Loan,   
 estimate = .pred\_Yes) %>%  
 autoplot()

 ### lift 커브

tree\_test\_pred %>%  
 lift\_curve(truth = Personal\_Loan,   
 estimate = .pred\_Yes) %>%  
 autoplot()

 ### 중요변수 확인

tree\_test\_fit %>%  
 pluck(".workflow", 1) %>%   
 extract\_fit\_parsnip() %>% # `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
 vip(num\_features = 20)

 ## 10.train, test 검정결과 비교 ### 정오분류표(confusion matrix) 비교

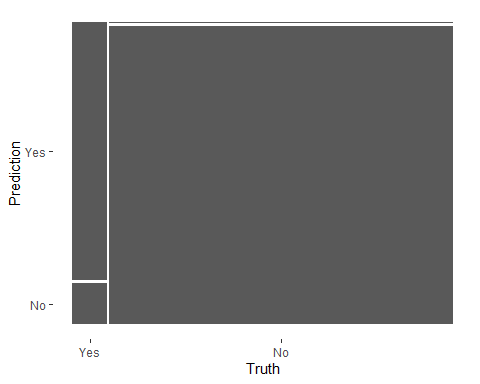
tree\_train\_conf

## Truth  
## Prediction Yes No  
## Yes 279 10  
## No 44 3167

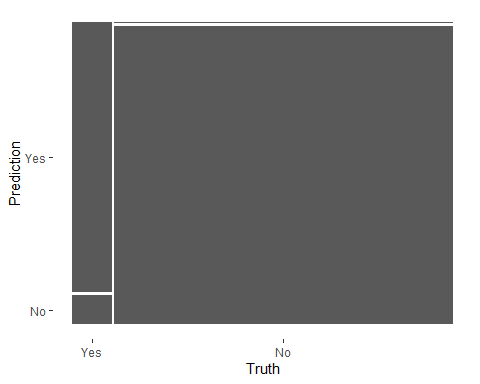
tree\_test\_conf

## Truth  
## Prediction Yes No  
## Yes 142 7  
## No 15 1336

autoplot(tree\_train\_conf, type = "mosaic") # mosaic



autoplot(tree\_test\_conf, type = "mosaic")

 ### 검정결과 비교

summary(tree\_train\_conf)

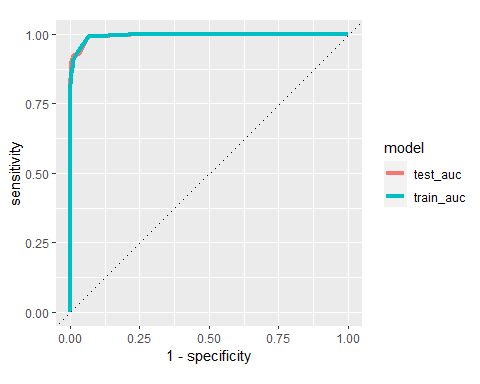
## # A tibble: 13 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 accuracy binary 0.985   
## 2 kap binary 0.903   
## 3 sens binary 0.864   
## 4 spec binary 0.997   
## 5 ppv binary 0.965   
## 6 npv binary 0.986   
## 7 mcc binary 0.905   
## 8 j\_index binary 0.861   
## 9 bal\_accuracy binary 0.930   
## 10 detection\_prevalence binary 0.0826  
## 11 precision binary 0.965   
## 12 recall binary 0.864   
## 13 f\_meas binary 0.912

summary(tree\_test\_conf)

## # A tibble: 13 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 accuracy binary 0.985   
## 2 kap binary 0.920   
## 3 sens binary 0.904   
## 4 spec binary 0.995   
## 5 ppv binary 0.953   
## 6 npv binary 0.989   
## 7 mcc binary 0.920   
## 8 j\_index binary 0.899   
## 9 bal\_accuracy binary 0.950   
## 10 detection\_prevalence binary 0.0993  
## 11 precision binary 0.953   
## 12 recall binary 0.904   
## 13 f\_meas binary 0.928

### ROC 커브 비교

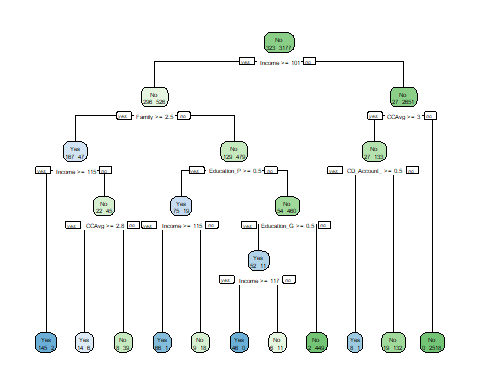
bind\_rows(train\_auc, test\_auc) %>%   
 ggplot(aes(x = 1 - specificity,   
 y = sensitivity,   
 color = model)) +   
 geom\_path(lwd = 1.5) +  
 geom\_abline(lty = 3) +   
 coord\_equal()

 ## 11.decision tree 만들기

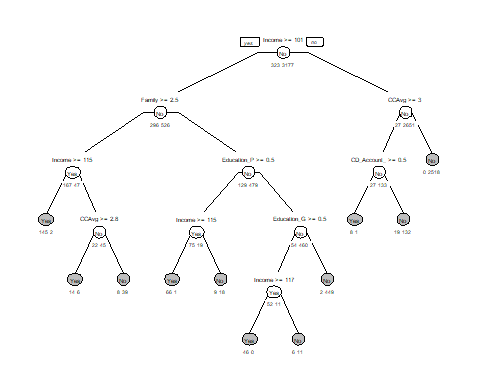
rpart\_fit <-   
 tree\_train\_fit %>%   
 extract\_fit\_parsnip() # pull\_workflow\_fit()` was deprecated in workflows 0.2.3.

### 모형 1

rpart.plot(x = rpart\_fit$fit,  
 yesno = 2,  
 type = 2,   
 extra = 1,   
 split.font = 1,   
 varlen = -10,  
 roundint=FALSE)

 ### 모형 2

prp(x = rpart\_fit$fit,   
 type = 1,   
 extra = 1,   
 under = TRUE,   
 split.font = 1,   
 varlen = -10,  
 box.col=ifelse(rpart\_fit$fit$frame$var == "<leaf>", 'gray', 'white'),  
 roundint=FALSE)

 help(prp) type = 나무그래프 표현 종류 extra= 추가 정보 표시, 1=노드의 관측수 표시 under= box 아래 관측값 표시, default=False split.font = 글자 font, default=2(bold) varlen = 변수이름 길이, default=-8, 0=full name box.col=box 색깔