Problem_1

Problem 1: Predicting healthcare charges, in USD, using a patient's age and BMI as features

```r
library(readr)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v stringr   1.5.1
## v forcats   1.0.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
↪  to become errors
```

```r
library(rpart) # this is what we use to make the decision tree
```

```
## Warning: package 'rpart' was built under R version 4.3.3
```

```r
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.3
```

```r
library(dplyr)
```

```r
df <- read_csv("./insurance_charges.csv")
```

```
## Rows: 1338 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (5): age, bmi, charges, f_bmi, cardiovascular_care_cost
##
## 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.
```

```r
head(df)
```

```
## # A tibble: 6 x 5
##     age   bmi charges f_bmi cardiovascular_care_cost
##   <dbl> <dbl>   <dbl> <dbl>                    <dbl>
## 1    19  3.90  16885. 0.591                    1876.
## 2    18  5.19   1726. 0.716                    2466.
```

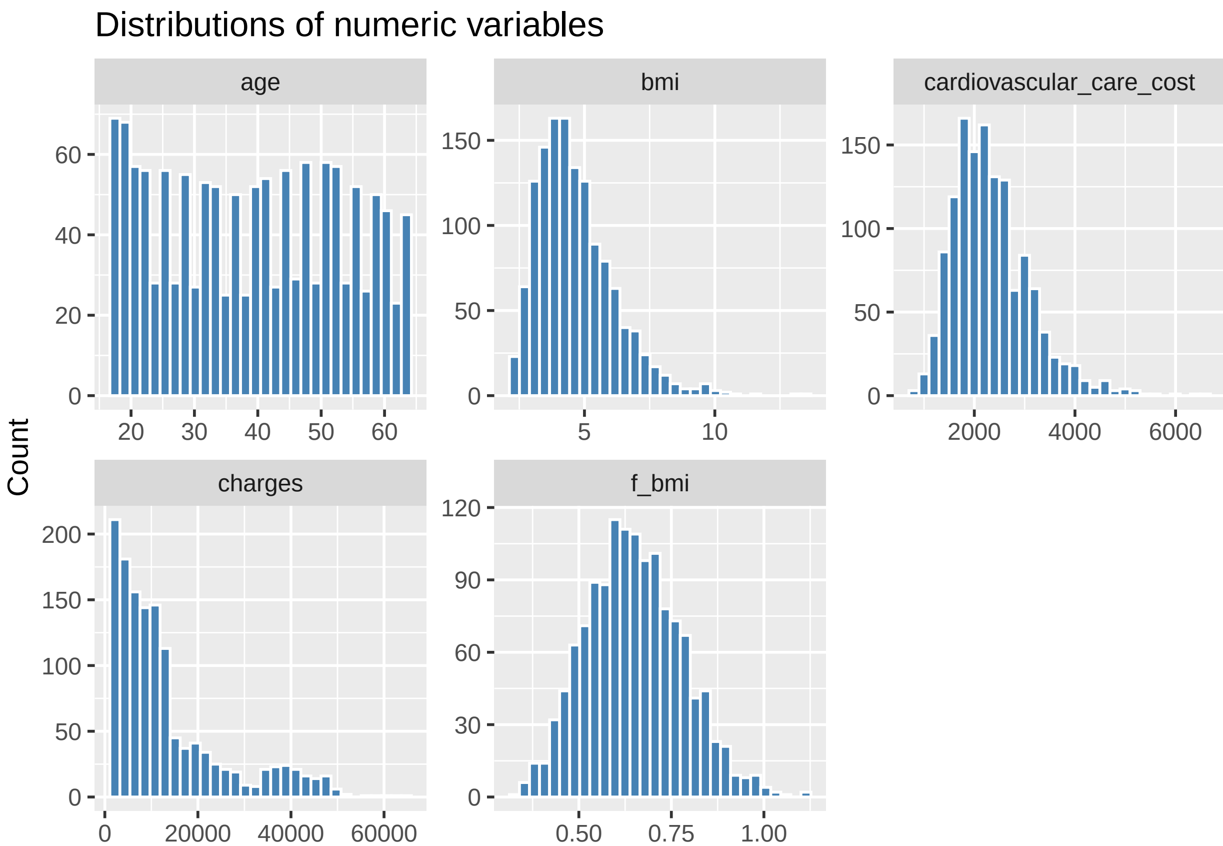
```
## 3    28  5.00   4449. 0.699                    2473.
## 4    33  3.03  21984. 0.481                    1656.
## 5    32  4.09   3867. 0.612                    1933.
## 6    31  3.51   3757. 0.545                    1548.
```

```r
summary(df)
```

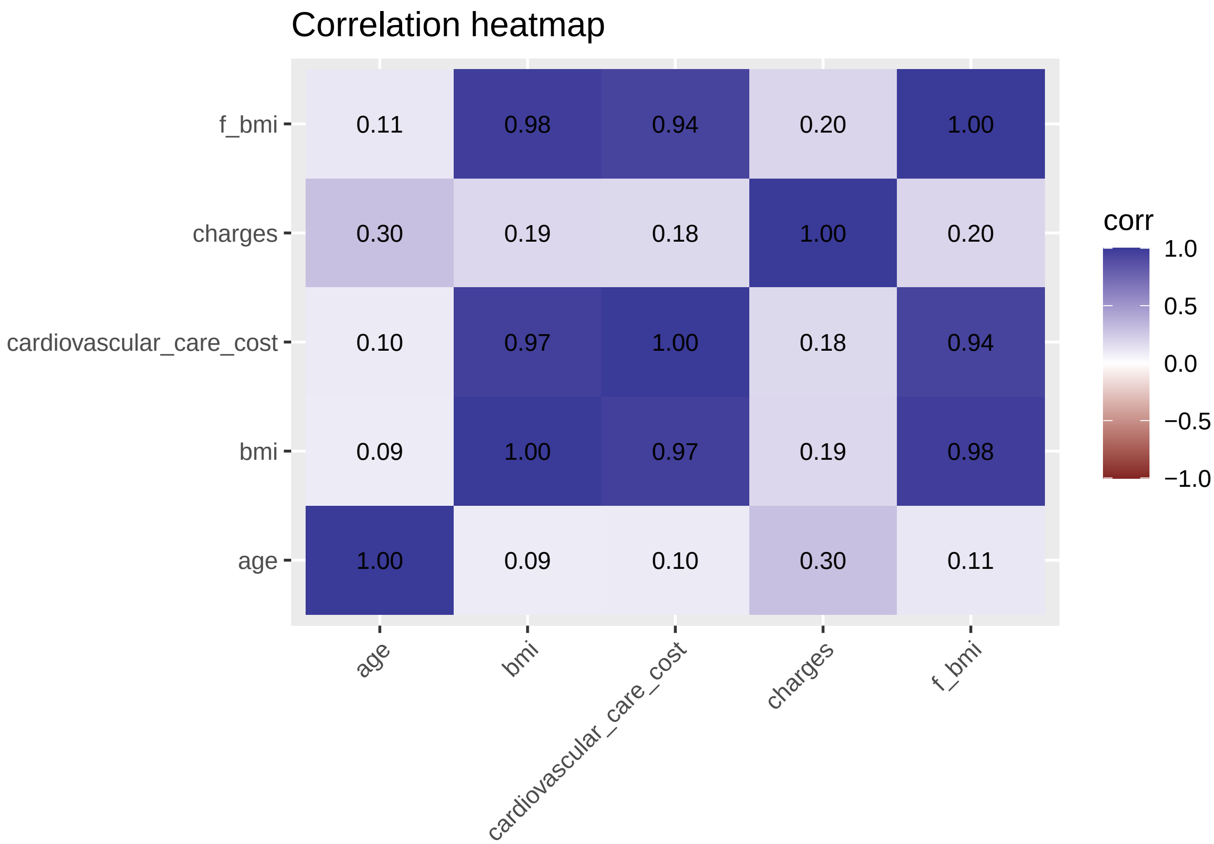
```
##       age             bmi            charges          f_bmi       
##  Min.   :18.00   Min.   : 2.179   Min.   : 1122   Min.   :0.3382  
##  1st Qu.:27.00   1st Qu.: 3.607   1st Qu.: 4740   1st Qu.:0.5572  
##  Median :39.00   Median : 4.407   Median : 9382   Median :0.6442  
##  Mean   :39.21   Mean   : 4.672   Mean   :13270   Mean   :0.6497  
##  3rd Qu.:51.00   3rd Qu.: 5.434   3rd Qu.:16640   3rd Qu.:0.7351  
##  Max.   :64.00   Max.   :13.359   Max.   :63770   Max.   :1.1258  
##  cardiovascular_care_cost
##  Min.   : 827.1          
##  1st Qu.:1784.1          
##  Median :2204.5          
##  Mean   :2338.0          
##  3rd Qu.:2720.7          
##  Max.   :6621.8          
```

Let us start with really basic exploratory data analysis to gain some understanding on our data.

```r
# Distributions of the numeric columns
df |>
  select(where(is.numeric)) |>
  pivot_longer(everything(), names_to = "var", values_to = "val") |>
  ggplot(aes(val)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "white") +
  facet_wrap(~ var, scales = "free") +
  labs(title = "Distributions of numeric variables", x = NULL, y = "Count")
```



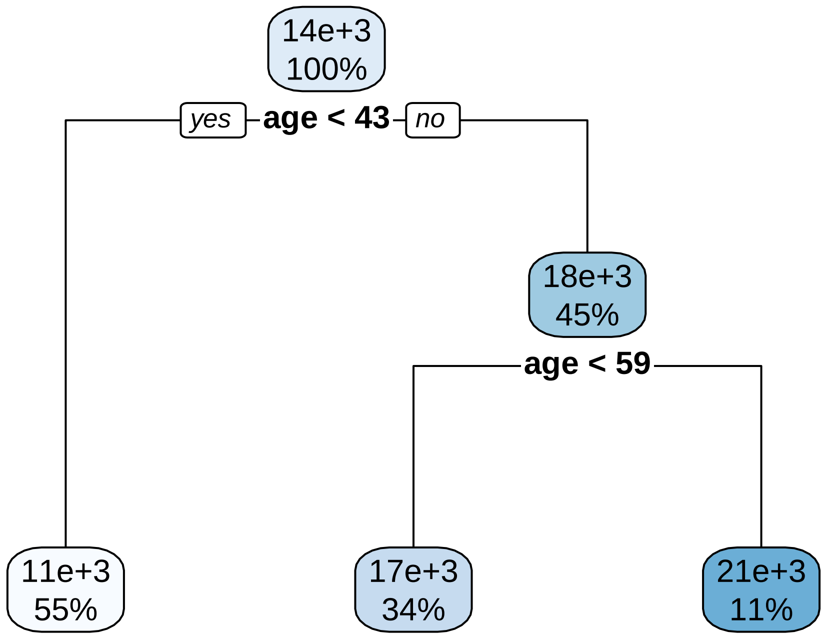

```r
# Correlation matrix heatmap
corr_mat <- cor(df |> select(where(is.numeric)), use = "pairwise.complete.obs")

corr_mat |>
  as.data.frame() |>
  tibble::rownames_to_column("var1") |>
  pivot_longer(-var1, names_to = "var2", values_to = "corr") |>
  ggplot(aes(var1, var2, fill = corr)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%.2f", corr)), size = 3) +
  scale_fill_gradient2(limits = c(-1, 1)) +
  labs(title = "Correlation heatmap", x = NULL, y = NULL) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



From the EDA, we can see that charges are right-skewed with a few high-cost outliers, while age is fairly uniform across the dataset. The correlation heatmap shows that BMI, f_bmi, and cardiovascular_care_cost are highly correlated with each other, but only moderately related to charges, suggesting potential multicollinearity among the BMI-related features.

a)

```r
# Decision tree predicting charges based on bmi and age.
model1 <- rpart(charges ~ age, bmi, data = df)

# Plots the decision tree
rpart.plot(model1)
```




```r
# Returns a summary of the model
summary(model1)
```

```
## Call:
## rpart(formula = charges ~ age, data = df, weights = bmi)
##   n= 1338
##
##           CP nsplit rel error    xerror       xstd
## 1 0.06646995      0 1.0000000 1.0020664 0.02247429
## 2 0.01095788      1 0.9335300 0.9376656 0.02146159
## 3 0.01000000      2 0.9225722 0.9351160 0.02155025
##
## Variable importance
## age
## 100
##
## Node number 1: 1338 observations,    complexity param=0.06646995
##   mean=13987.49, MSE=1.686433e+08
##   left son=2 (755 obs) right son=3 (583 obs)
##   Primary splits:
##       age < 42.5 to the left,  improve=0.06646995, (0 missing)
##
## Node number 2: 755 observations
##   mean=10961.62, MSE=1.548827e+08
##
## Node number 3: 583 observations,    complexity param=0.01095788
##   mean=17692.11, MSE=1.605566e+08
##   left son=6 (444 obs) right son=7 (139 obs)
##   Primary splits:
##       age < 58.5 to the left,  improve=0.02560143, (0 missing)
##
## Node number 6: 444 observations
##   mean=16548.01, MSE=1.542857e+08
```

```
##
## Node number 7: 139 observations
##   mean=21284.86, MSE=1.632305e+08
```

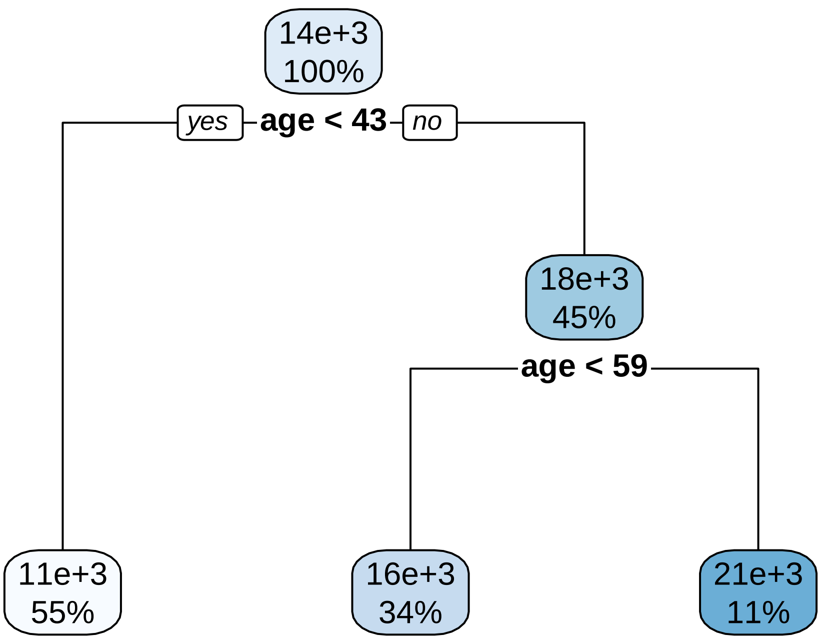
```r
# R-squared ( from training data)
pred <- predict(model1, df)
rss <- sum((df$charges - pred)^2)               # residual sum of squares
tss <- sum((df$charges - mean(df$charges))^2)   # total sum of squares
rsq <- 1 - rss/tss

cat("The Training R-squared is:", round(rsq, 3), "\n")
```

```
## The Training R-squared is: 0.088
```

The R-squared obtained is equal to 8.8%, which is low and shows that the model has not learnt deeply the relationship we want to model.

b)

```r
# Decision tree predicting charges based on f_bmi and age.
model2 <- rpart(charges ~ age, f_bmi, data = df)

# Plots the decision tree
rpart.plot(model2)
```




```r
# Returns a summary of the model
summary(model2)
```

```
## Call:
## rpart(formula = charges ~ age, data = df, weights = f_bmi)
##   n= 1338
##
##           CP nsplit rel error    xerror       xstd
## 1 0.07042646      0 1.0000000 1.0009390 0.06147059
```

```
## 2 0.01190059      1 0.9295735 0.9318000 0.05859990
## 3 0.01000000      2 0.9176729 0.9335435 0.05908664
##
## Variable importance
## age
## 100
##
## Node number 1: 1338 observations,    complexity param=0.07042646
##   mean=13747.75, MSE=1.606751e+08
##   left son=2 (755 obs) right son=3 (583 obs)
##   Primary splits:
##       age < 42.5 to the left,  improve=0.07042646, (0 missing)
##
## Node number 2: 755 observations
##   mean=10733.14, MSE=1.463161e+08
##
## Node number 3: 583 observations,    complexity param=0.01190059
##   mean=17501.4, MSE=1.531487e+08
##   left son=6 (444 obs) right son=7 (139 obs)
##   Primary splits:
##       age < 58.5 to the left,  improve=0.02803182, (0 missing)
##
## Node number 6: 444 observations
##   mean=16332.09, MSE=1.456117e+08
##
## Node number 7: 139 observations
##   mean=21172.83, MSE=1.590411e+08
```

```r
# R-squared ( from training data)
pred <- predict(model2, df)
rss <- sum((df$charges - pred)^2)                # residual sum of squares
tss <- sum((df$charges - mean(df$charges))^2)   # total sum of squares
rsq <- 1 - rss/tss

cat("The Training R-squared is:", round(rsq, 3), "\n")
```

```
## The Training R-squared is: 0.09
```

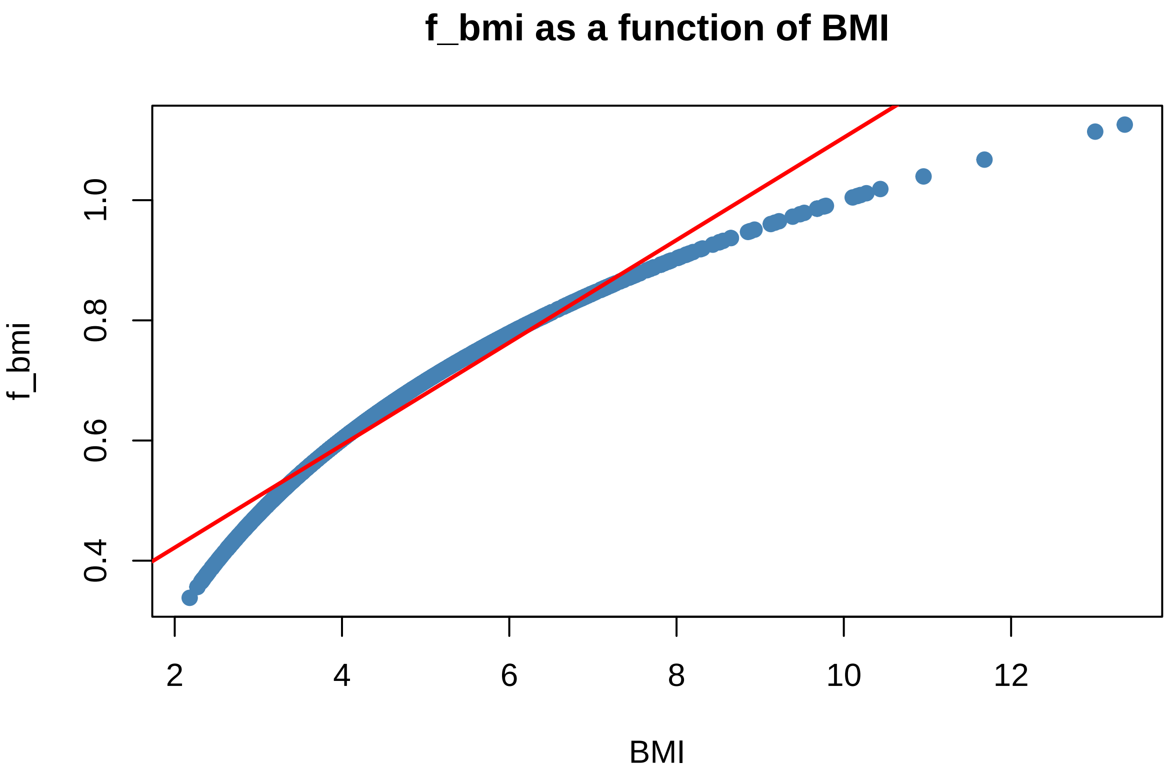
We get a R-squared value fairly small (9%), which testifies that the model has not well learnt the training data and that perhaps, we need a more complex model.

c)

From the trees' plots performed using rpart, we can see that the structures are exactly the same and the cutoffs are too (age < 43, and age < 59 were the variables chosen). Addtionally, the R-squared obtained from model1 and model2 are very close (9% and 8.8%, respectively). This hints on the high correlation between f_bmi and bmi. As we can see from the correlation heatmap of our EDA, the f_bmi and bmi features are almost perfectly correlated (Dark blue) with a coefficient of 0.98! On top of this, f_bmi as a function of bmi shows that f_bmi is a nonlinear transformation of bmi, likely a scaled or polynomial/log-type function that compresses larger BMI values, since the relationship is strongly positive but clearly curved rather than linear.

```r
# Plot f_bmi as a function of bmi
plot(df$bmi, df$f_bmi,
     xlab = "BMI",
```

```
     ylab = "f_bmi",
     main = "f_bmi as a function of BMI",
     pch = 19, col = "steelblue")
abline(lm(f_bmi ~ bmi, data = df), col = "red", lwd = 2)
```



d)

e)

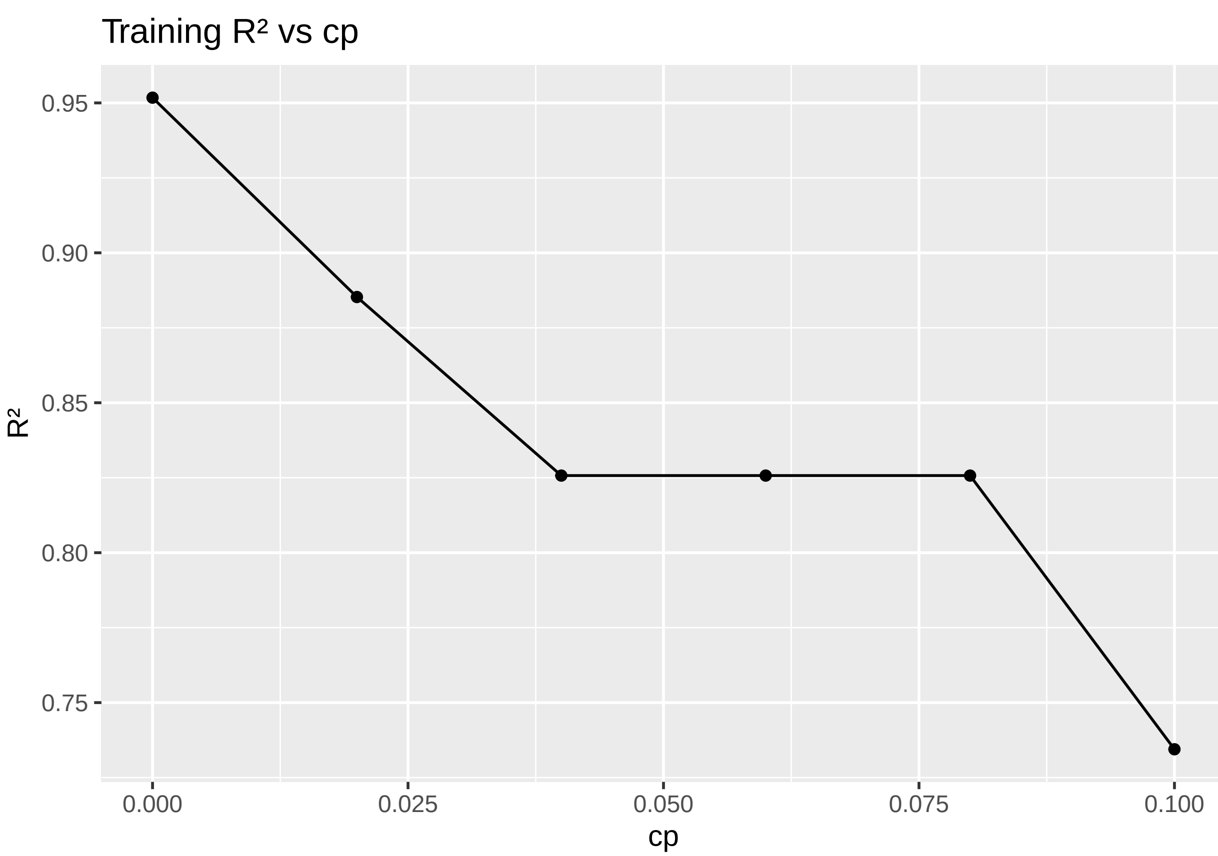
Firstly, we should define clearly which dependent variables we would like to keep. We will not use bmi since it is highly correlated to f_bmi, as shown above. We will keep age, and add charges, since it is not a value we aim to predict anymore.

```
# Grid of cp values
cps <- c(0, 0.02, 0.04, 0.06, 0.08, 0.1)

# Fit tree at a given cp and compute training metrics
fit_one <- function(cp) {

  fit <- rpart(cardiovascular_care_cost ~ age + f_bmi + charges,
              data = df,
              control = rpart.control(cp = cp))  # keep other defaults

  y <- df$cardiovascular_care_cost
  pred <- predict(fit, df)
  rss <- sum((y - pred)^2); tss <- sum((y - mean(y))^2)
  r2  <- 1 - rss/tss
  rmse <- sqrt(mean((y - pred)^2))

  leaves <- sum(fit$frame$var == "<leaf>")
```

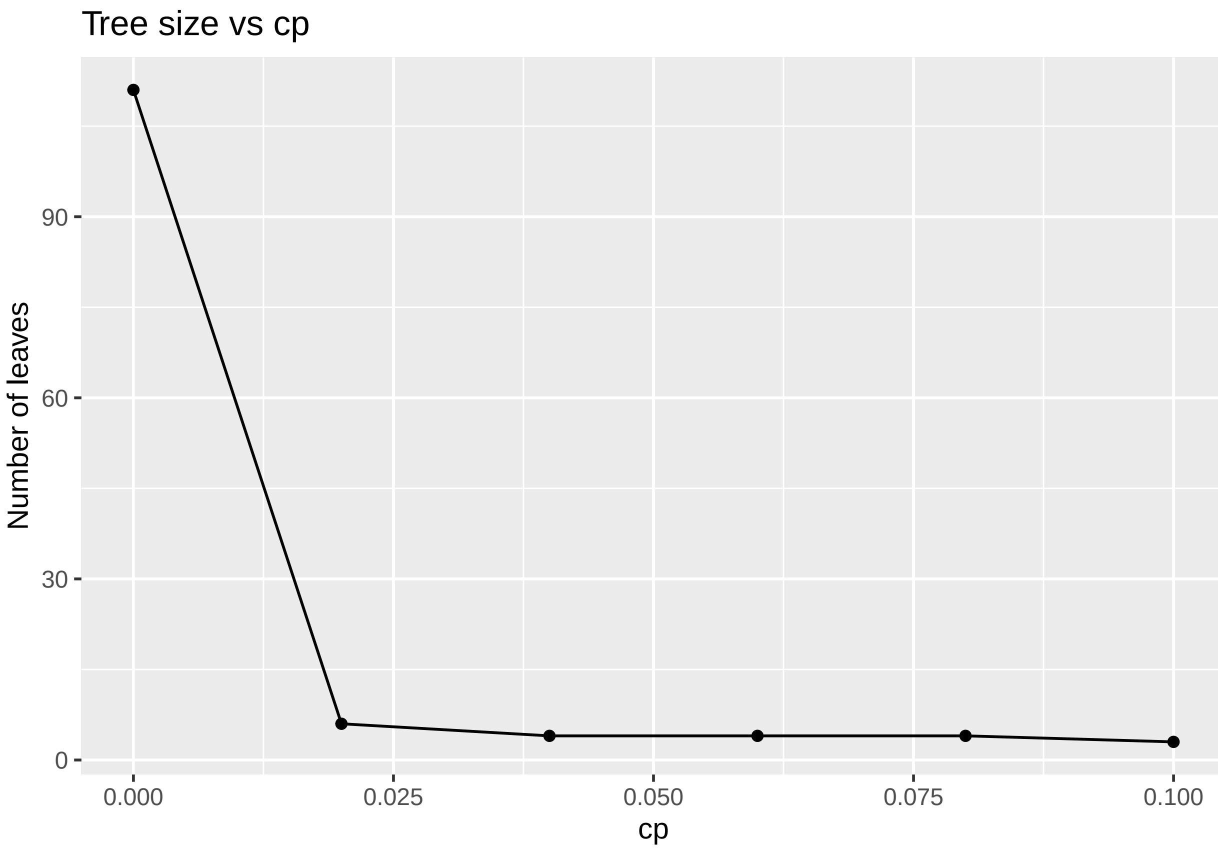
```r
  depth <- max(rpart:::tree.depth(as.numeric(row.names(fit$frame))))

  tibble(cp = cp, R2 = r2, RMSE = rmse, Leaves = leaves, Depth = depth, model =
   ↪  list(fit))
}

# Run fits
res <- bind_rows(lapply(cps, fit_one))

# Show summary table (metrics only)
res %>% select(cp, R2, RMSE, Leaves, Depth)
```

```
## # A tibble: 6 x 5
##      cp    R2  RMSE Leaves Depth
##   <dbl> <dbl> <dbl>  <int> <dbl>
## 1  0    0.952  169.    111    12
## 2  0.02 0.885  261.      6     3
## 3  0.04 0.826  321.      4     2
## 4  0.06 0.826  321.      4     2
## 5  0.08 0.826  321.      4     2
## 6  0.1  0.734  397.      3     2
```

```r
# Plots how fit and complexity change with cp
ggplot(res, aes(cp, R2)) + geom_line() + geom_point() +
  labs(title = "Training R² vs cp", x = "cp", y = "R²")
```



```r
ggplot(res, aes(cp, Leaves)) + geom_line() + geom_point() +
  labs(title = "Tree size vs cp", x = "cp", y = "Number of leaves")
```

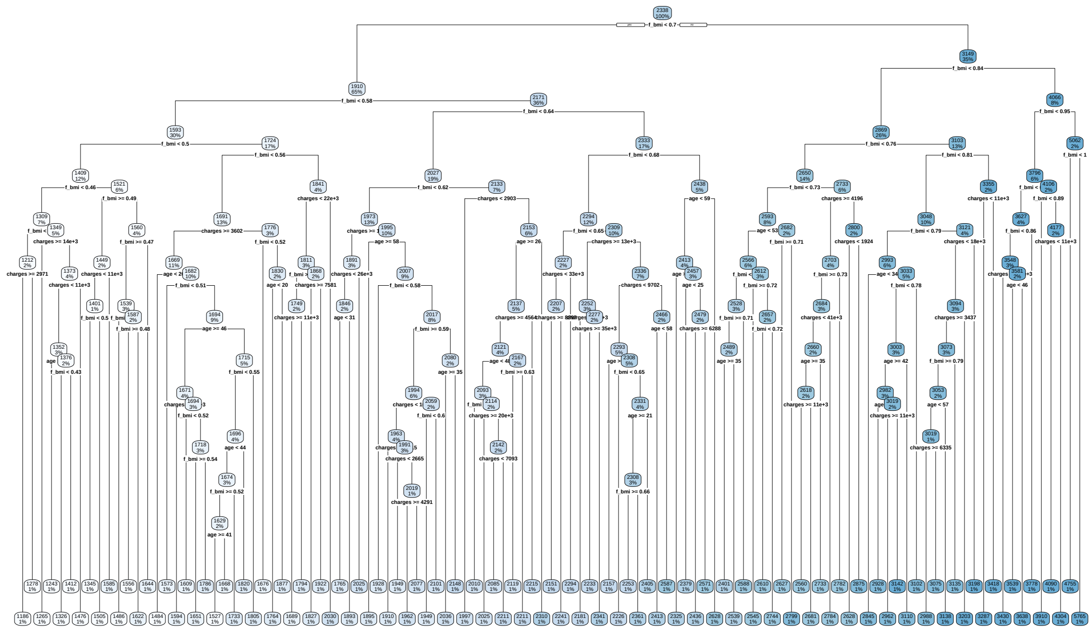



```r
# Plot each tree (one after another)
for (i in seq_len(nrow(res))) {
  cat("\n\n## cp =", res$cp[i], "\n")
  rpart.plot(res$model[[i]],
             main = paste0("CART: cardiovascular_care_cost (cp = ", res$cp[i], ")"))
}
```

```
##
##
## ## cp = 0
```

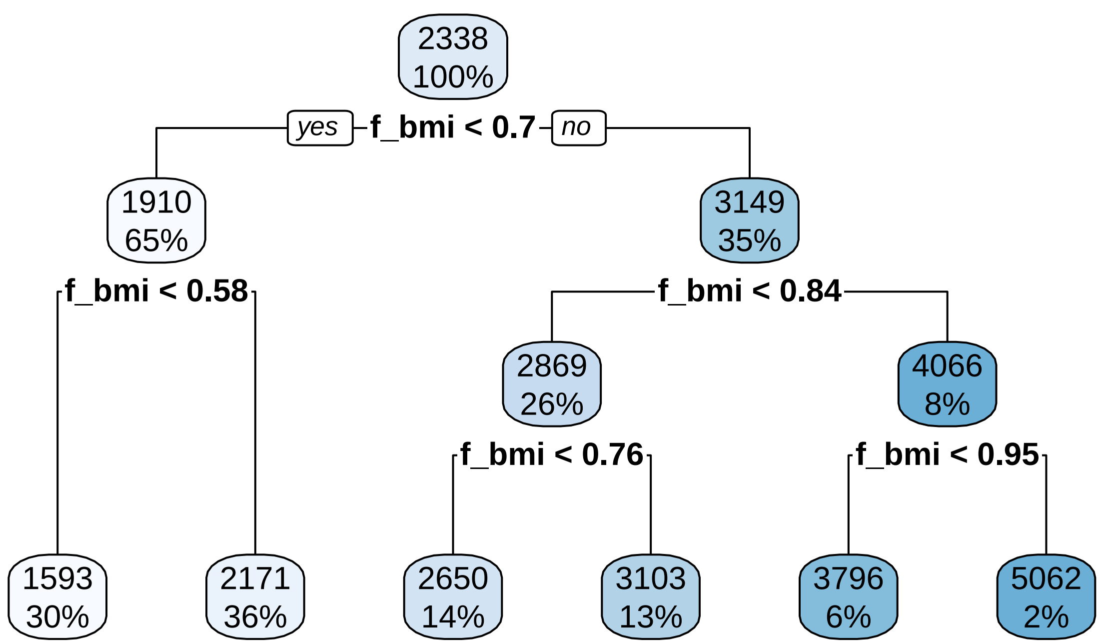
```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```

CART: cardiovascular_care_cost (cp = 0)



```
##
##
## ## cp = 0.02
```

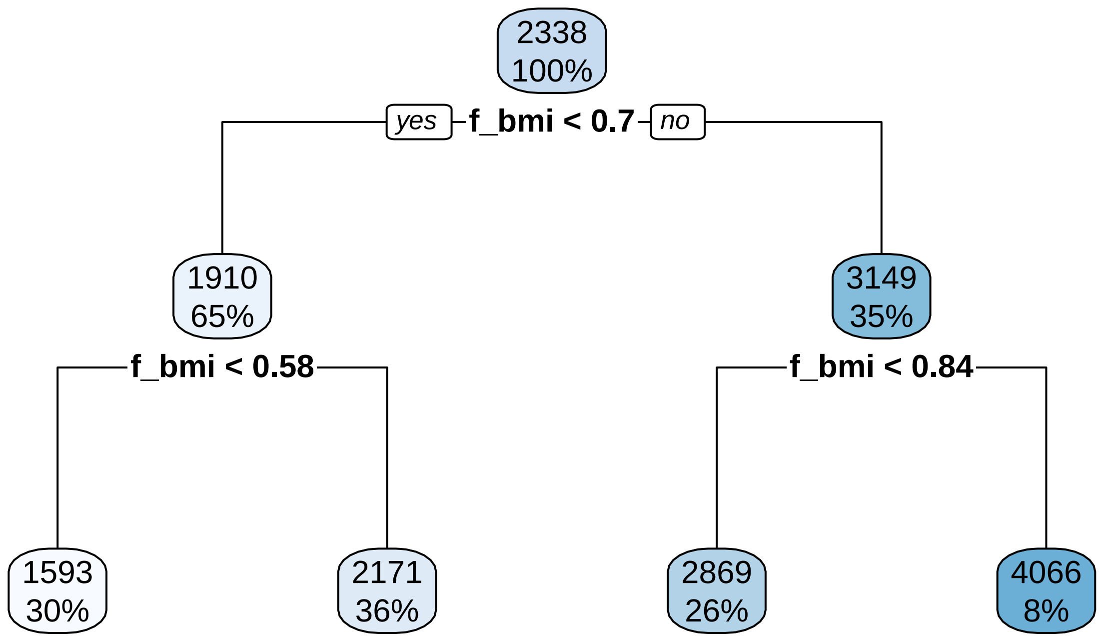
CART: cardiovascular_care_cost (cp = 0.02)




```
##
##
```

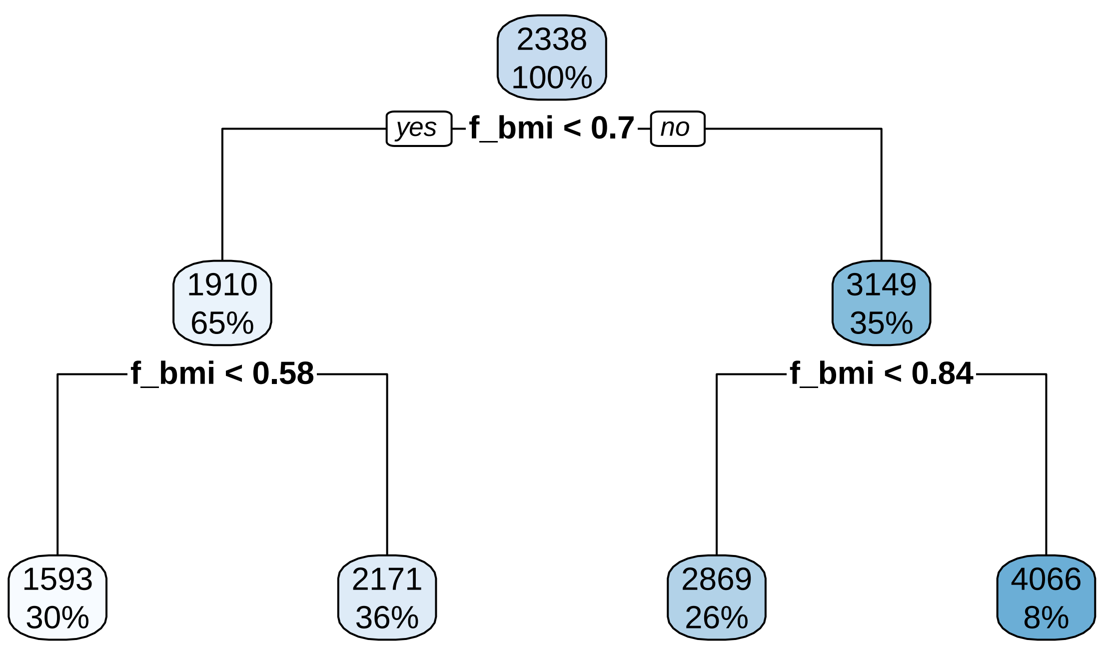
```
## ## cp = 0.04
```

CART: cardiovascular_care_cost (cp = 0.04)

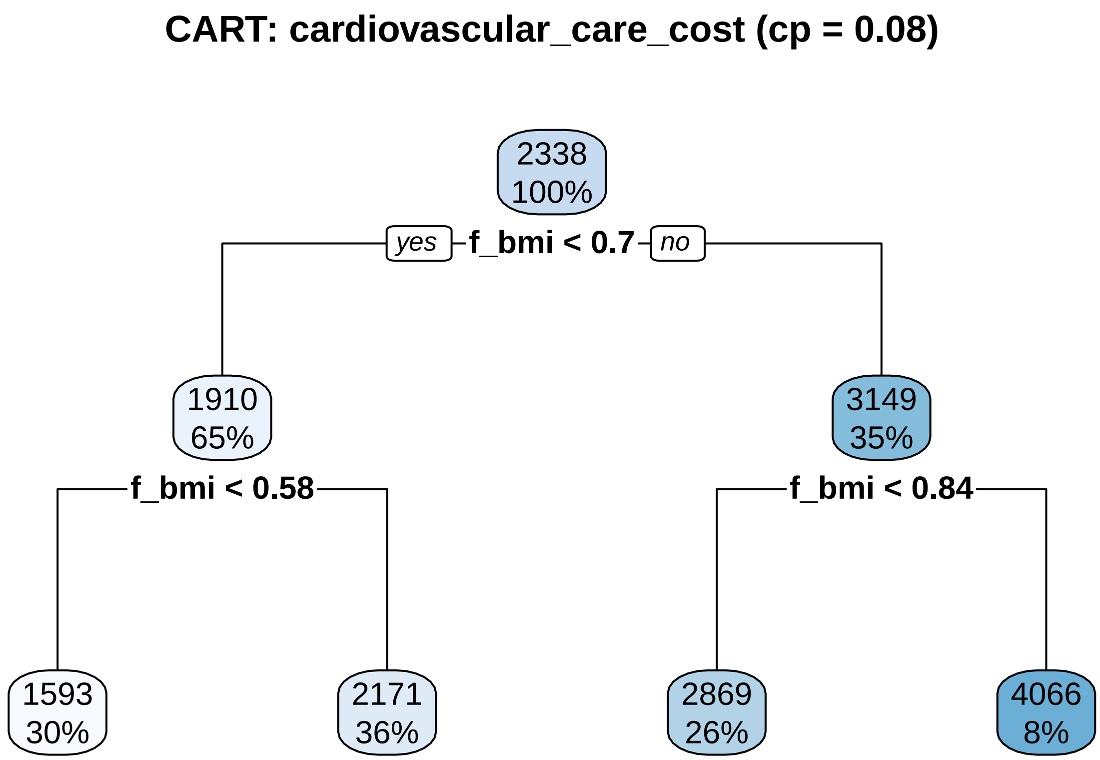



```
##
##
## ## cp = 0.06
```

CART: cardiovascular_care_cost (cp = 0.06)

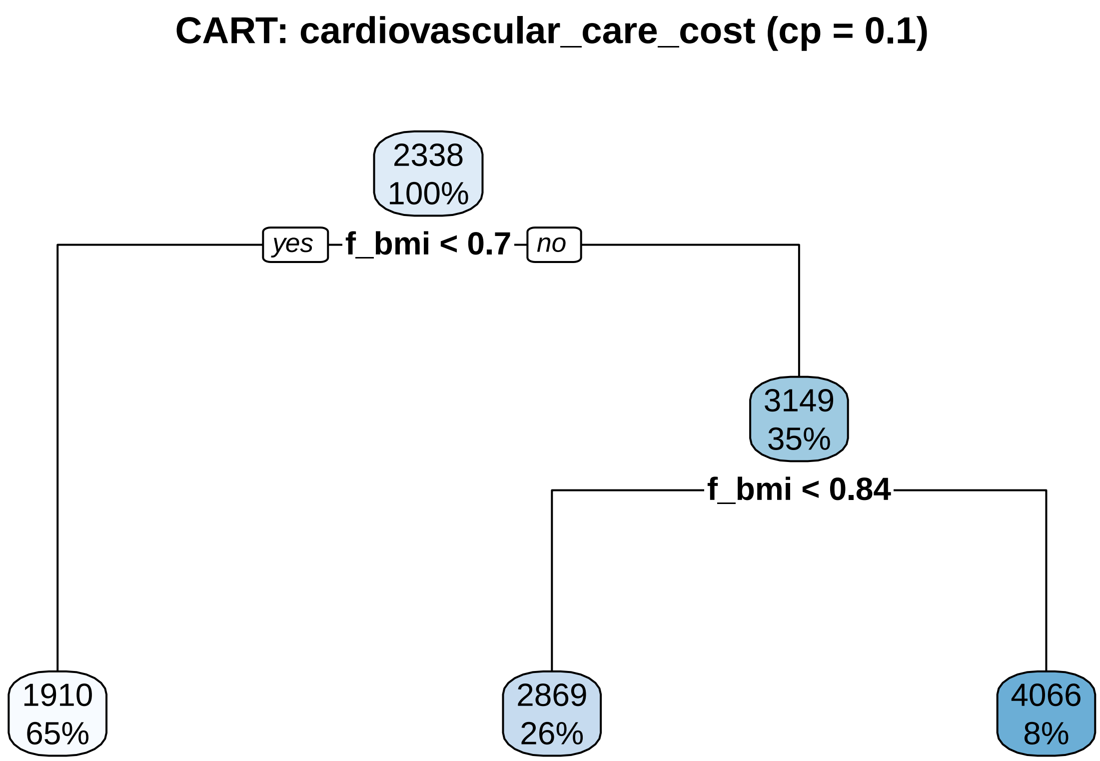


```
##
##
## ## cp = 0.08
```



CART: cardiovascular_care_cost (cp = 0.08)
2338
100%
yes
f_bmi < 0.7
no
1910
65%
3149
35%
f_bmi < 0.58
f_bmi < 0.84
1593
30%
2171
36%
2869
26%
4066
8%


```
##
##
## ## cp = 0.1
```



When cp = 0, the tree grows very deep (12 levels, 111 leaves), leading to extremely high training R^2 (0.95) but clear overfitting, as seen in the highly complex structure. Increasing cp prunes the tree aggressively: at cp = 0.02, the tree shrinks to just 6 leaves and R^2 drops to 0.89. For cp = 0.04–0.08, the tree stabilizes with 4 leaves and depth 2, achieving R^2 around 0.83, which indicates a simpler but still reasonable fit. Finally, at cp = 0.10, the tree becomes even smaller (3 leaves, depth 2) and the R^2 falls further to 0.73, showing underfitting.

In summary, smaller cp values allow more complex trees with higher apparent training accuracy but risk overfitting, while larger cp values lead to simpler trees with reduced variance but higher bias. This trade-off highlights the importance of tuning cp to balance model complexity and predictive power, based on needs for expanability (e.g. Shallow tree is preferred as it can be visualized and human-parsed quickly) or performance, for instance.