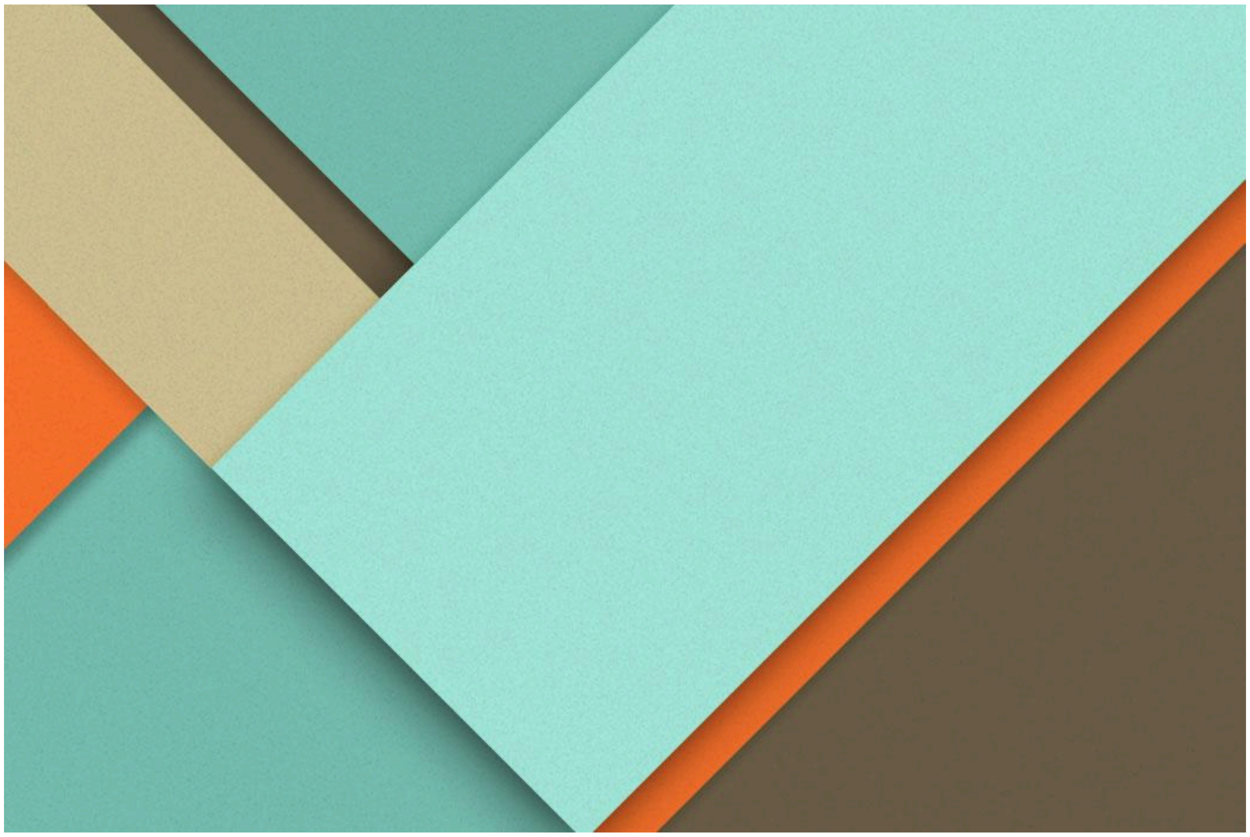


Deliverable 2

09.29.2025

—

M14

Riya Parikh

Hindy Rossignol

Ioannis Panagiotopoulos

Mrugank Pednekar

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     1.3.0
## 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 error
```

```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)
```

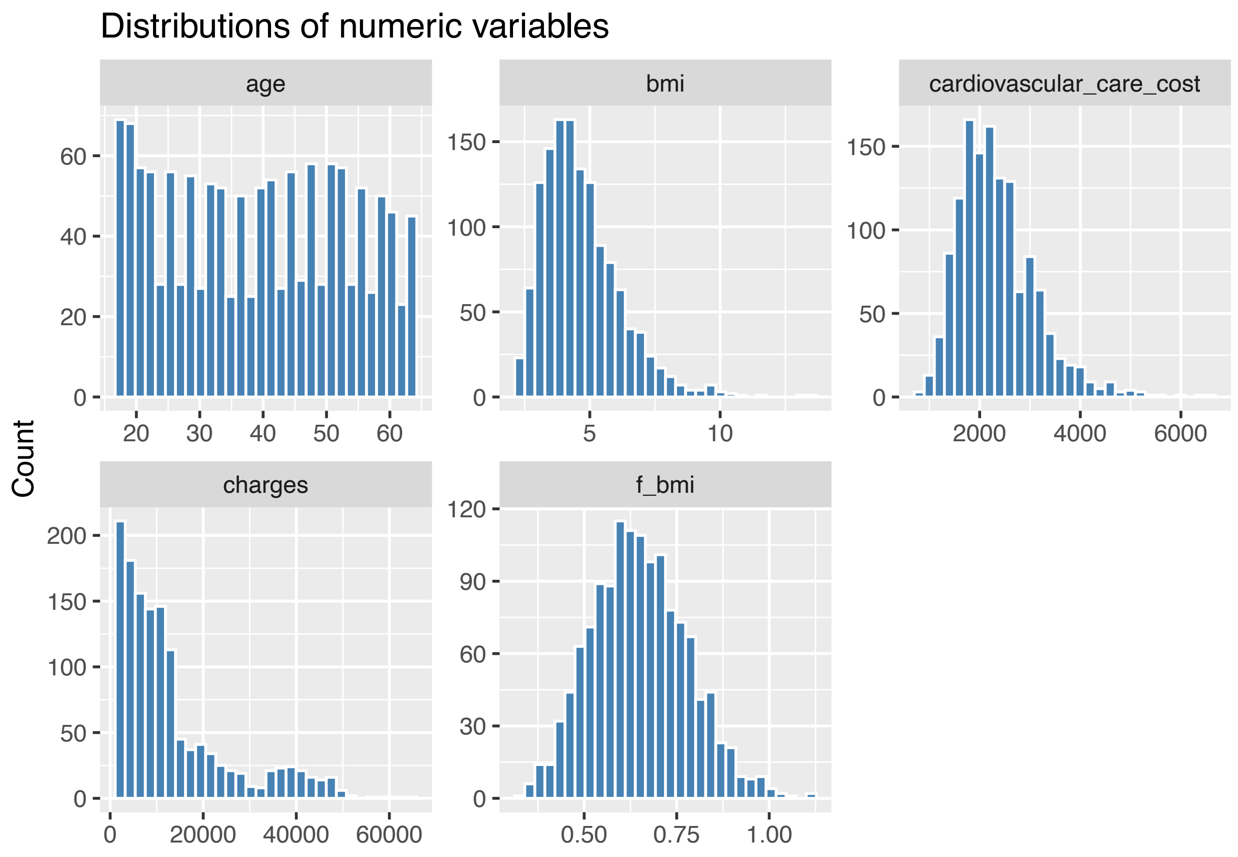
```
## # A tibble: 6 x 5
##     age   bmi charges f_bmi cardiovascular_care_cost
##   <dbl> <dbl>   <dbl> <dbl>                    <dbl>
## 1    19  3.90  16885. 0.591                    1876.
## 2    18  5.19   1726. 0.716                    2466.
## 3    28  5.00   4449. 0.699                    2473.
## 4    33  3.03  21984. 0.481                    1656.
## 5    32  4.09   3867. 0.612                        1933.
## 6    31  3.51   3757. 0.545                        1548.
```

```
summary(df)
```

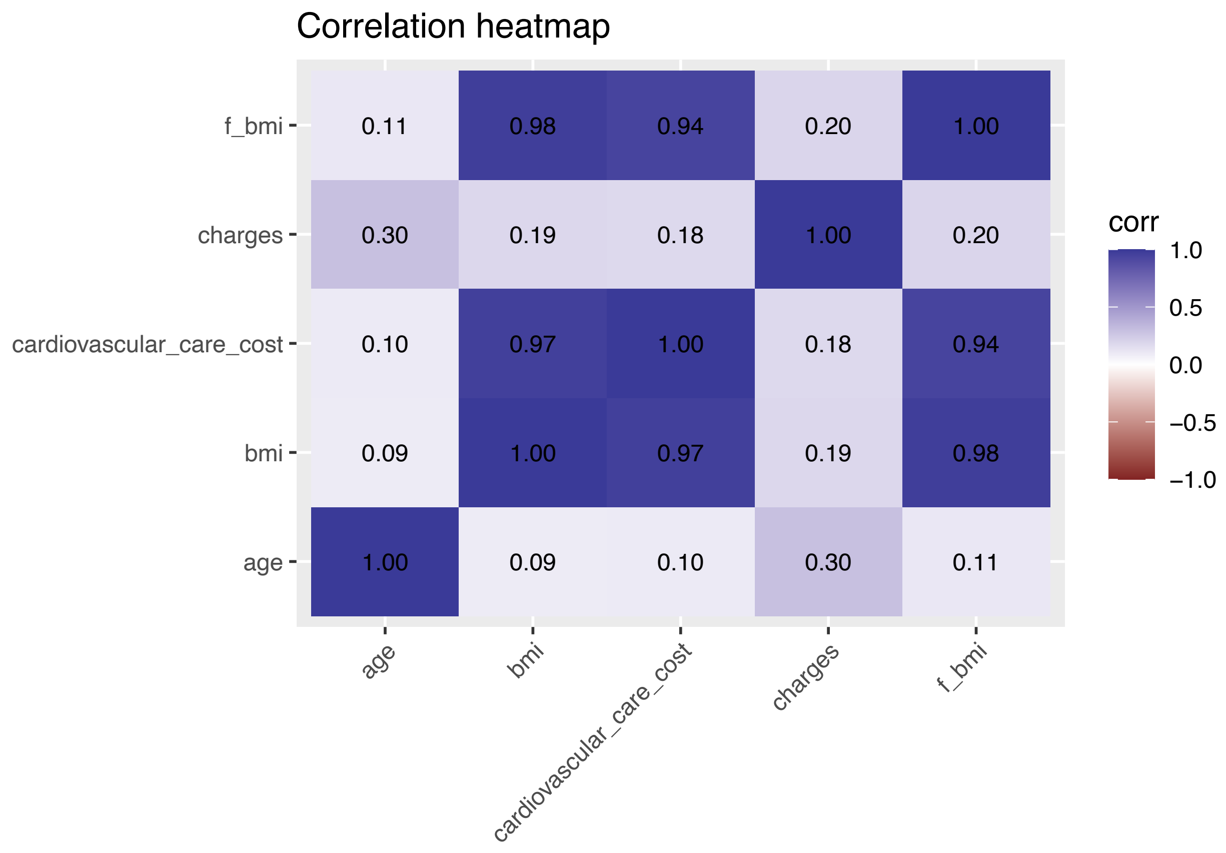
```
##       age             bmi            charges          f_bmi       
##  Min.   :18.00   Min.   : 2.179   Min.   : 1122   Min.   :0.3382  
##  1st Qu.:27.00   1st Qu.: 3.607   1st Qu.: 4740   1st Qu.:0.5572  
##  Median :39.00   Median : 4.407   Median : 9382   Median :0.6442  
##  Mean   :39.21   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.

```
# 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")
```



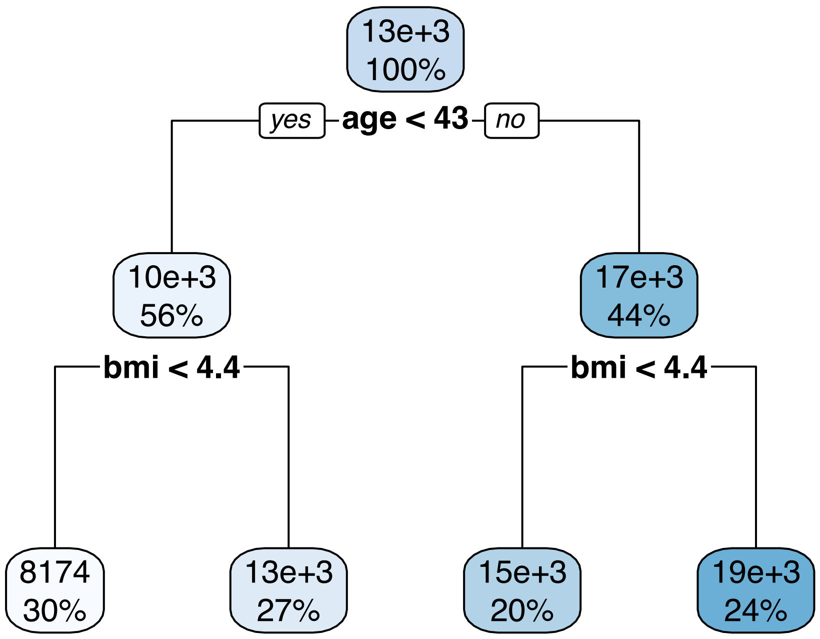
```
# Correlation matrix heatmap
corr_mat <- cor(df |> select(where(is.numeric)), use = "pairwise.complete.obs")

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 and f_bmi are almost perfectly correlated (~0.98), while their relationship to charges is only moderate. Because CART models split on orderings, monotone transforms like f_bmi usually yield nearly identical trees to BMI.

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 + bmi, data = df)
##   n= 1338
##
##           CP nsplit rel error    xerror       xstd
## 1 0.07793137      0 1.0000000 1.0025356 0.05189781
## 2 0.01920458      1 0.9220686 0.9289775 0.04950382
## 3 0.01507422      2 0.9028640 0.9435374 0.04787011
## 4 0.01000000      3 0.8877898 0.9264762 0.04549575
##
## Variable importance
## age bmi
##  68  32
##
## Node number 1: 1338 observations,    complexity param=0.07793137
##   mean=13270.42, MSE=1.465428e+08
##   left son=2 (755 obs) right son=3 (583 obs)
##   Primary splits:
##       age < 42.5     to the left,  improve=0.07793137, (0 missing)
##       bmi < 4.357944 to the left,  improve=0.04212369, (0 missing)
##   Surrogate splits:
##       bmi < 5.728587 to the left,  agree=0.582, adj=0.041, (0 split)
##
```

```
## Node number 2: 755 observations,    complexity param=0.01920458
##   mean=10300.81, MSE=1.313497e+08
##   left son=4 (396 obs) right son=5 (359 obs)
##   Primary splits:
##       bmi < 4.357944 to the left,  improve=0.03797077, (0 missing)
##       age < 26.5     to the left,  improve=0.01289901, (0 missing)
##   Surrogate splits:
##       age < 18.5     to the right, agree=0.534, adj=0.019, (0 split)
##
## Node number 3: 583 observations,    complexity param=0.01507422
##   mean=17116.14, MSE=1.400084e+08
##   left son=6 (262 obs) right son=7 (321 obs)
##   Primary splits:
##       bmi < 4.392632 to the left,  improve=0.03621035, (0 missing)
##       age < 58.5     to the left,  improve=0.03075757, (0 missing)
##   Surrogate splits:
##       age < 47.5     to the left,  agree=0.566, adj=0.034, (0 split)
##
## Node number 4: 396 observations
##   mean=8174.444, MSE=5.228144e+07
##
## Node number 5: 359 observations
##   mean=12646.34, MSE=2.080781e+08
##
## Node number 6: 262 observations
##   mean=14623.87, MSE=5.261606e+07
##
## Node number 7: 321 observations
##   mean=19150.33, MSE=2.021303e+08
```

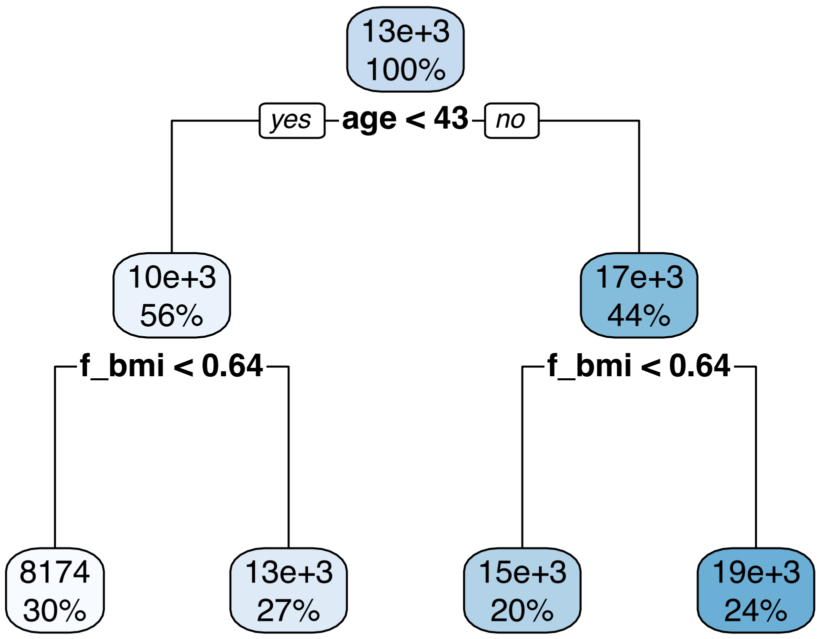
```r
# R-squared ( from training data)
pred <- predict(model1, df)
rss <- sum((df$charges - pred)^2)                # residual sum of squares
tss <- sum((df$charges - mean(df$charges))^2)   # total sum of squares
rsq <- 1 - rss/tss

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

```
## The Training R-squared is: 0.112
```

The training R^2 is ~11.2%, and the cross-validated relative error from printcp is ~0.92, meaning the tree only modestly improves over predicting the mean. Age contributes most of the predictive power, with importance ~68 compared to 32 for BMI.

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)
```



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

```
## Call:
## rpart(formula = charges ~ age + f_bmi, data = df)
##   n= 1338
##
##           CP nsplit rel error    xerror       xstd
## 1 0.07793137      0 1.0000000 1.0024098 0.05200392
## 2 0.01920458      1 0.9220686 0.9268229 0.04920849
## 3 0.01507422      2 0.9028640 0.9253720 0.04590948
## 4 0.01000000      3 0.8877898 0.9227373 0.04552839
##
## Variable importance
##   age f_bmi
##    68    32
##
## Node number 1: 1338 observations,    complexity param=0.07793137
##   mean=13270.42, MSE=1.465428e+08
##   left son=2 (755 obs) right son=3 (583 obs)
##   Primary splits:
##       age   < 42.5      to the left,  improve=0.07793137, (0 missing)
##       f_bmi < 0.6392812 to the left,  improve=0.04212369, (0 missing)
##   Surrogate splits:
##       f_bmi < 0.7580472 to the left,  agree=0.582, adj=0.041, (0 split)
##
```

```
## Node number 2: 755 observations,    complexity param=0.01920458
##   mean=10300.81, MSE=1.313497e+08
##   left son=4 (396 obs) right son=5 (359 obs)
##   Primary splits:
##       f_bmi < 0.6392812 to the left,  improve=0.03797077, (0 missing)
##       age   < 26.5      to the left,  improve=0.01289901, (0 missing)
##   Surrogate splits:
##       age < 18.5      to the right, agree=0.534, adj=0.019, (0 split)
##
## Node number 3: 583 observations,    complexity param=0.01507422
##   mean=17116.14, MSE=1.400084e+08
##   left son=6 (262 obs) right son=7 (321 obs)
##   Primary splits:
##       f_bmi < 0.6427244 to the left,  improve=0.03621035, (0 missing)
##       age   < 58.5      to the left,  improve=0.03075757, (0 missing)
##   Surrogate splits:
##       age < 47.5      to the left,  agree=0.566, adj=0.034, (0 split)
##
## Node number 4: 396 observations
##   mean=8174.444, MSE=5.228144e+07
##
## Node number 5: 359 observations
##   mean=12646.34, MSE=2.080781e+08
##
## Node number 6: 262 observations
##   mean=14623.87, MSE=5.261606e+07
##
## Node number 7: 321 observations
##   mean=19150.33, MSE=2.021303e+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: 0.112
```

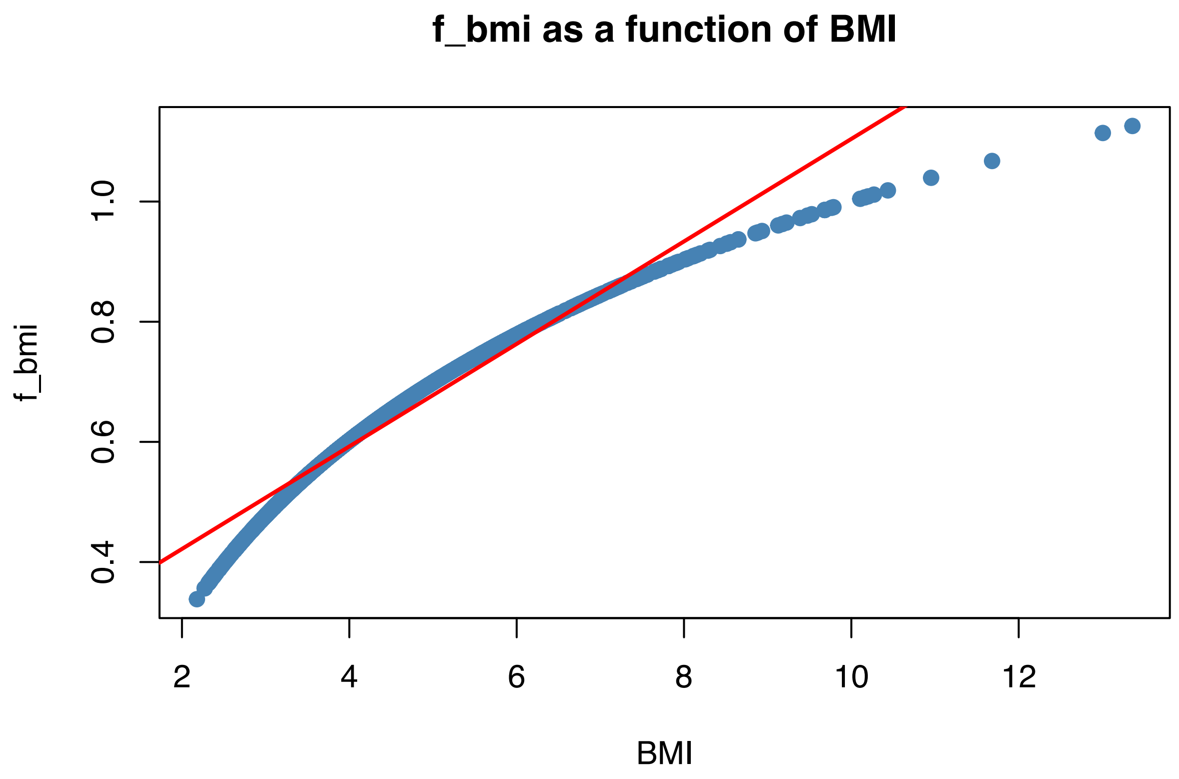
The training R^2 is again ~11.2%. The tree structure and splits are almost identical to model (a), which is expected since f_bmi is a monotone transform of BMI. Age still dominates in variable importance (~68 vs 32)

c)

Both trees have nearly identical splits (root split on age < 42.5, secondary splits on BMI ~ 4.36 or f_bmi ~ 0.64). Their training R^2 values are essentially the same (~11.2%). This outcome is expected because f_bmi is a near-monotone transform of BMI (correlation ~0.98), which preserves ordering. CART models base splits on order thresholds, so any monotone transform will yield similar partitions and performance.

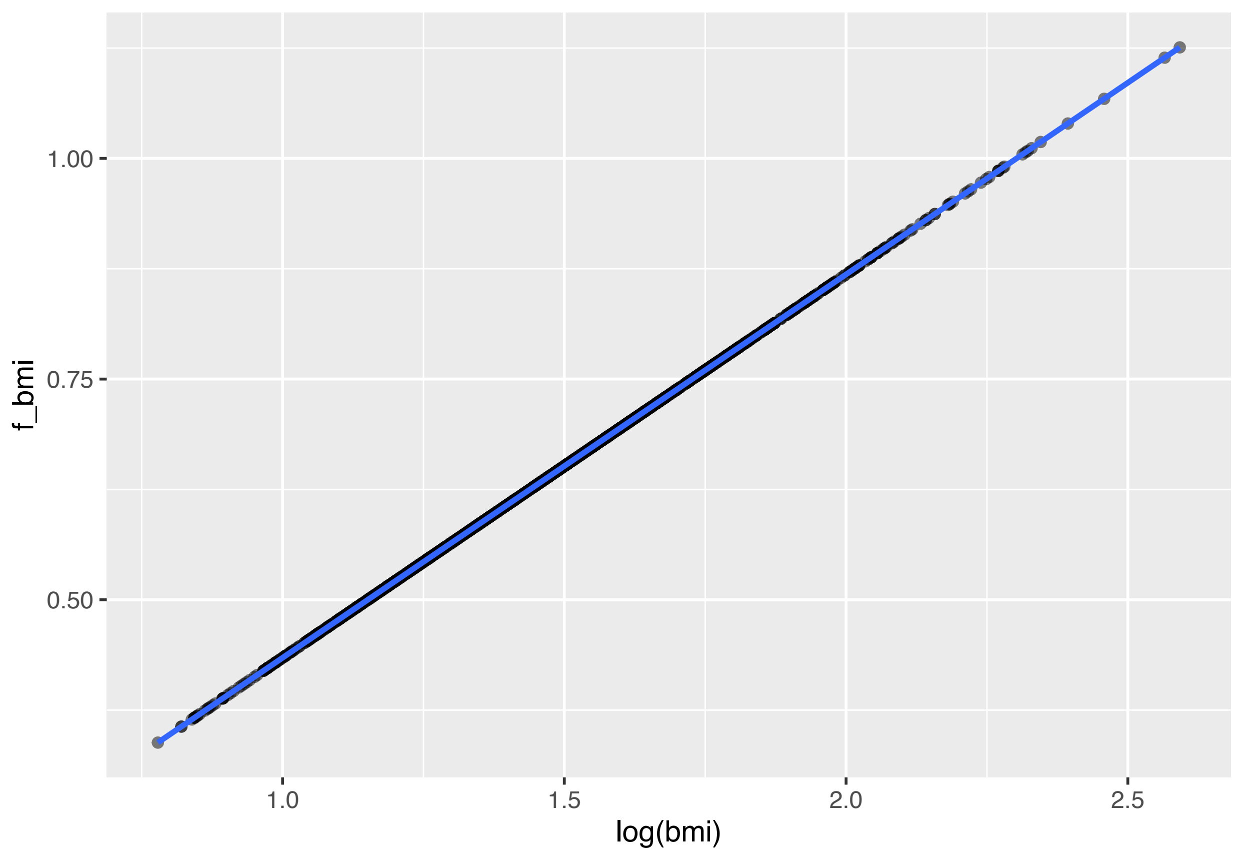
```r
# Plot f_bmi as a function of bmi
plot(df$bmi, df$f_bmi,
     xlab = "BMI",
     ylab = "f_bmi",
     main = "f_bmi as a function of BMI",
     pch = 19, col = "steelblue")
```

```r
abline(lm(f_bmi ~ bmi, data = df), col = "red", lwd = 2)
```



```r
ggplot(df, aes(log(bmi), f_bmi)) + geom_point(alpha=0.5) + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



> Here we can see that bmi vs f_bmi is related as f_bmi is a monotonic transform of bmi(scaling, log, square), therefore the trees will have similar R^2 (0.1122102) and comparable split logic although at different threshold values due to the difference in the scaling. Structural differences (number/depth of splits) may be minor; performance tends to stay close because decision trees rely primarily on ordering of values.

d)

> If f_bmi = g(bmi) where g is strictly monotone, then bmi_i < bmi_j <=> g(bmi_i) < g(bmi_j)
>
> CART chooses split thresholds by sorting a feature and scanning cut points; only the order matters. A strictly monotonic transform preserves that order, so the same partitions of the data are available (just at transformed thresholds). Hence training fit and structure remain essentially unchanged (up to ties/rounding). The entropy or impurity of the partitions remain the same as the proportion of points in each bucket doesnt change.

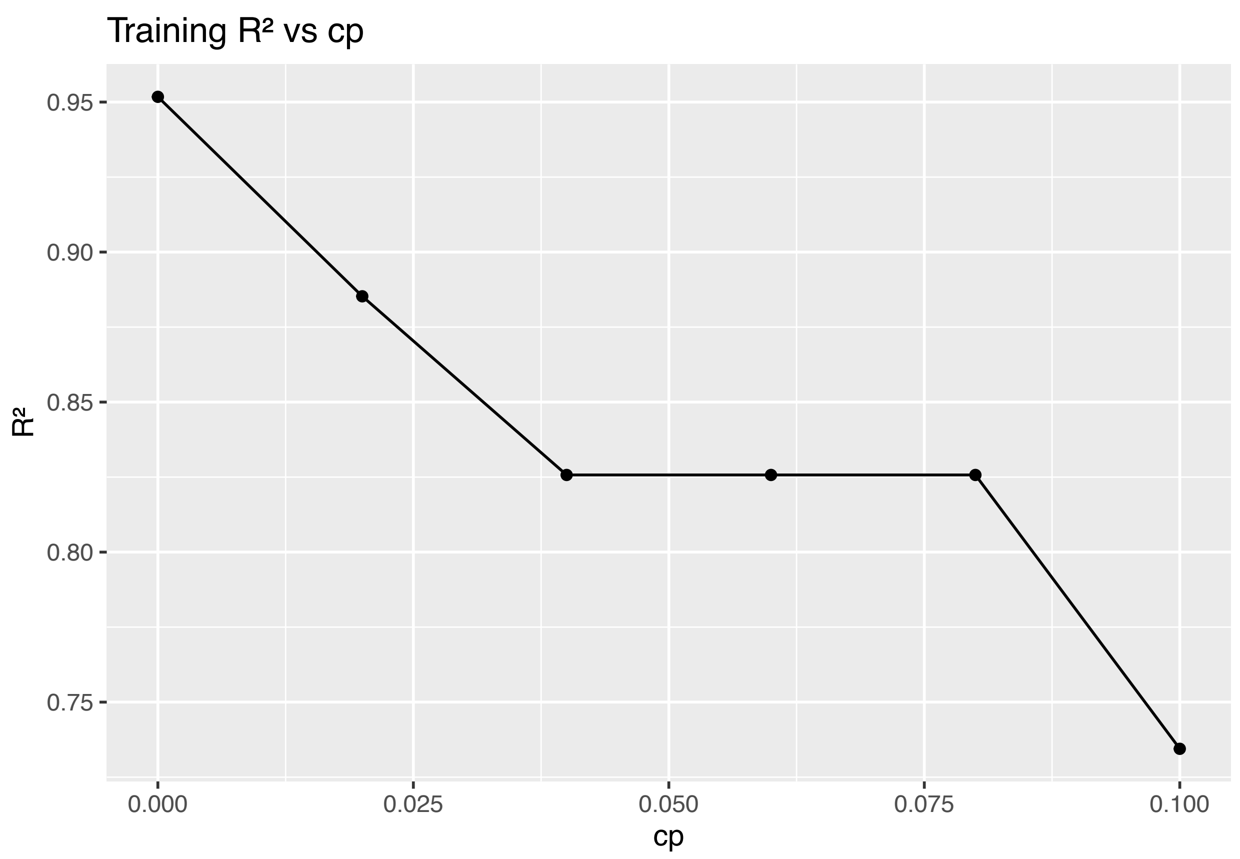
e)

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.

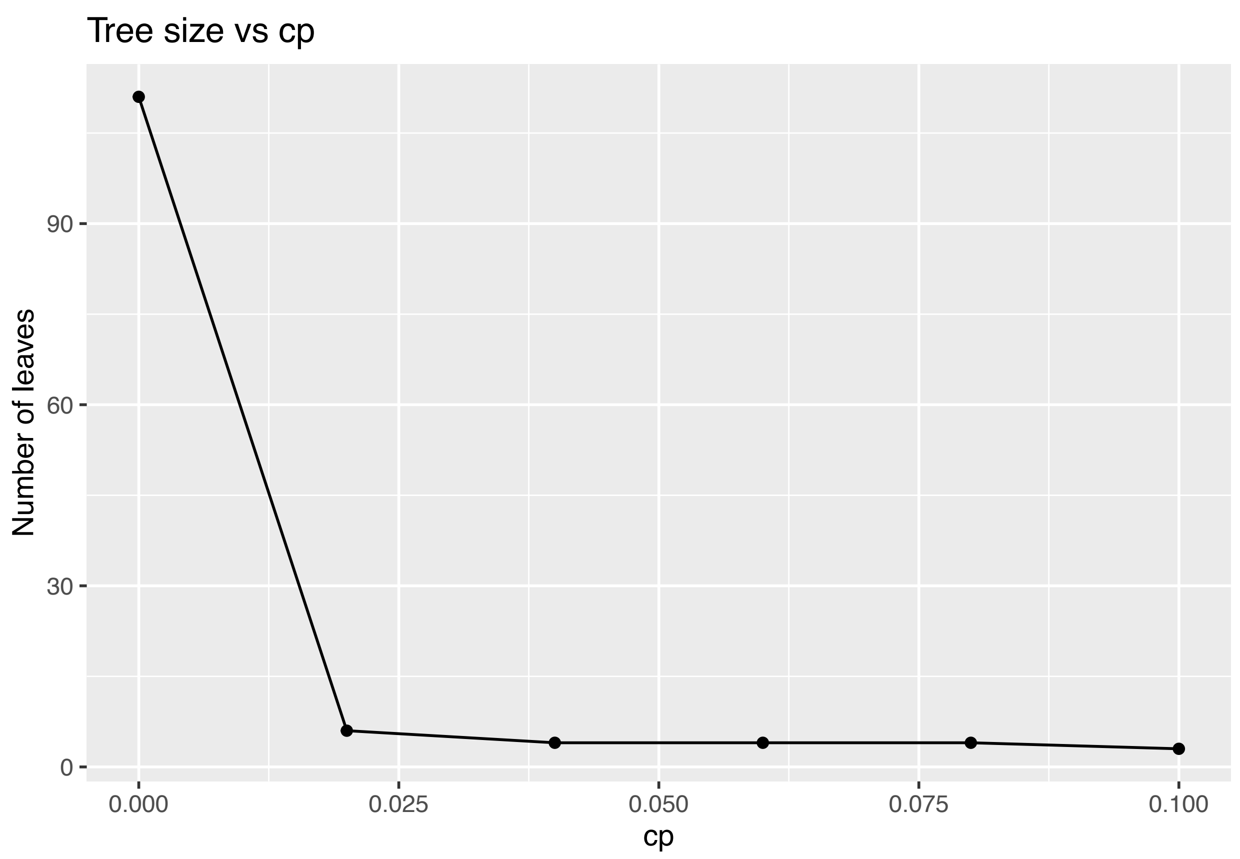
```r
# Grid of cp values
cps <- c(0, 0.02, 0.04, 0.06, 0.08, 0.1)

# 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>")

  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")
```

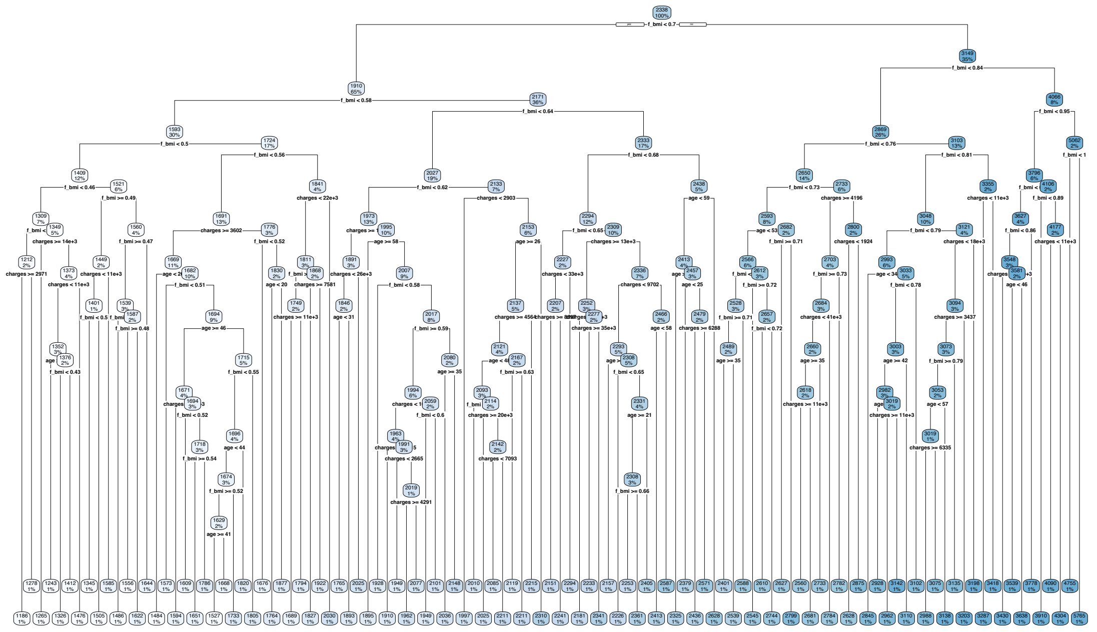


```r
# Plot each tree (one after another)
for (i in seq_len(nrow(res))) {
  cat("\n\n## cp =", res$cp[i], "\n")
  rpart.plot(res$model[[i]],
             main = paste0("CART: cardiovascular_care_cost (cp = ", res$cp[i], ")"))
}
```

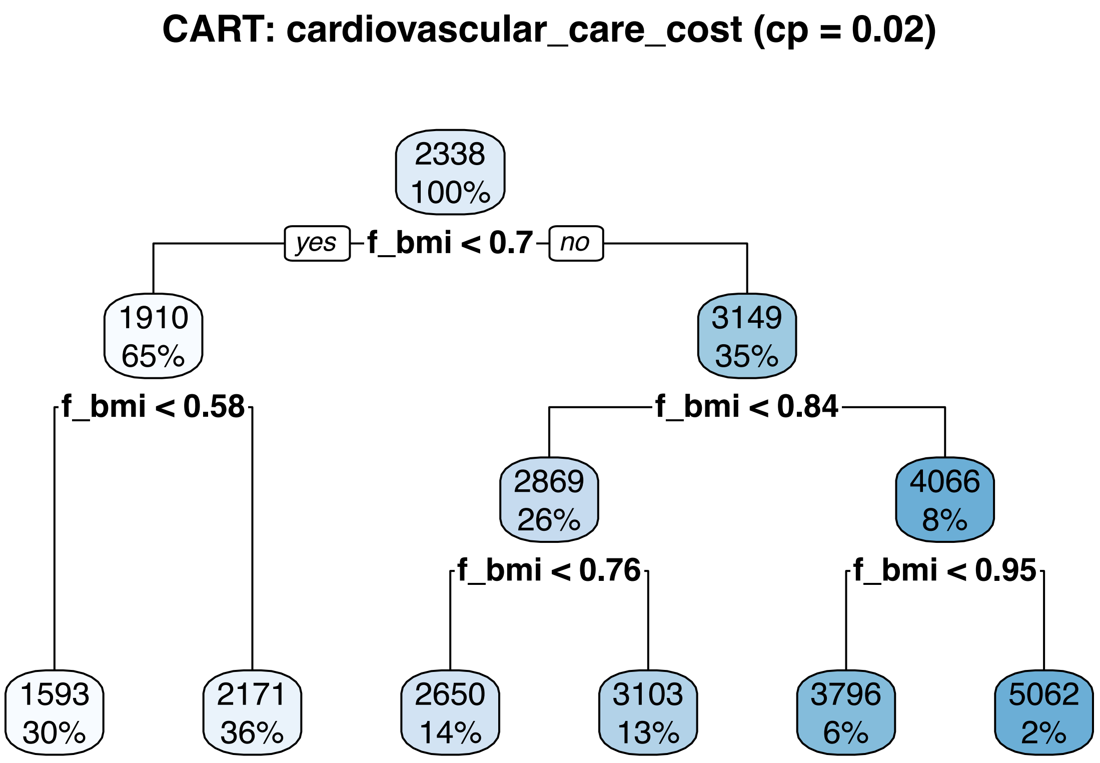
```
##
##
## ## cp = 0
```

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

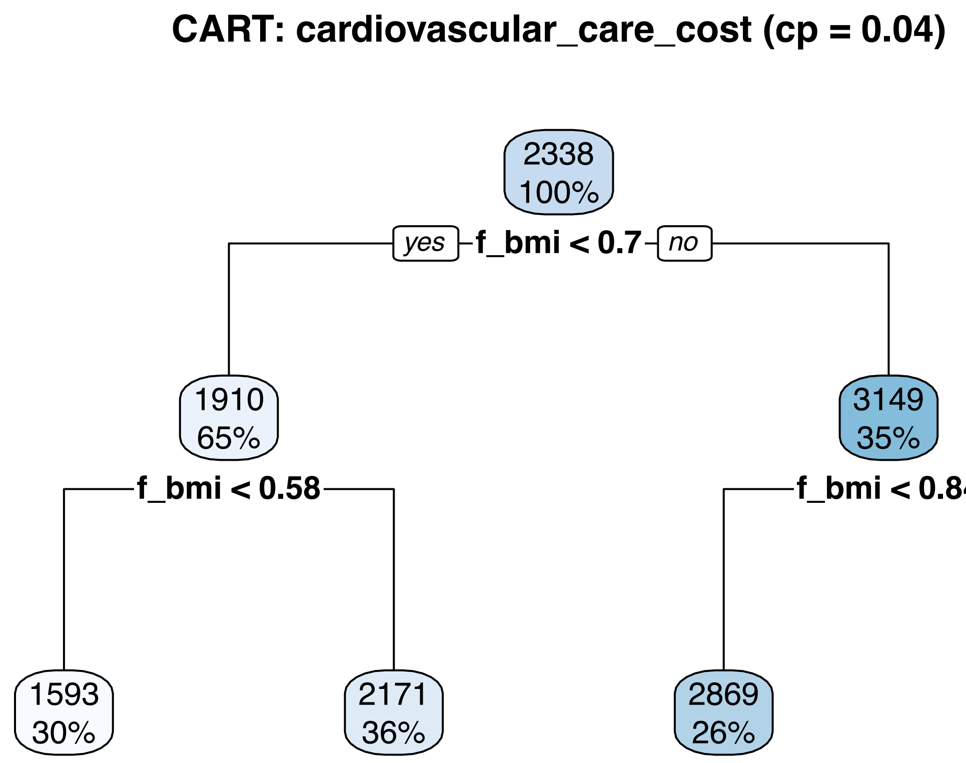
CART: cardiovascular_care_cost (cp = 0)



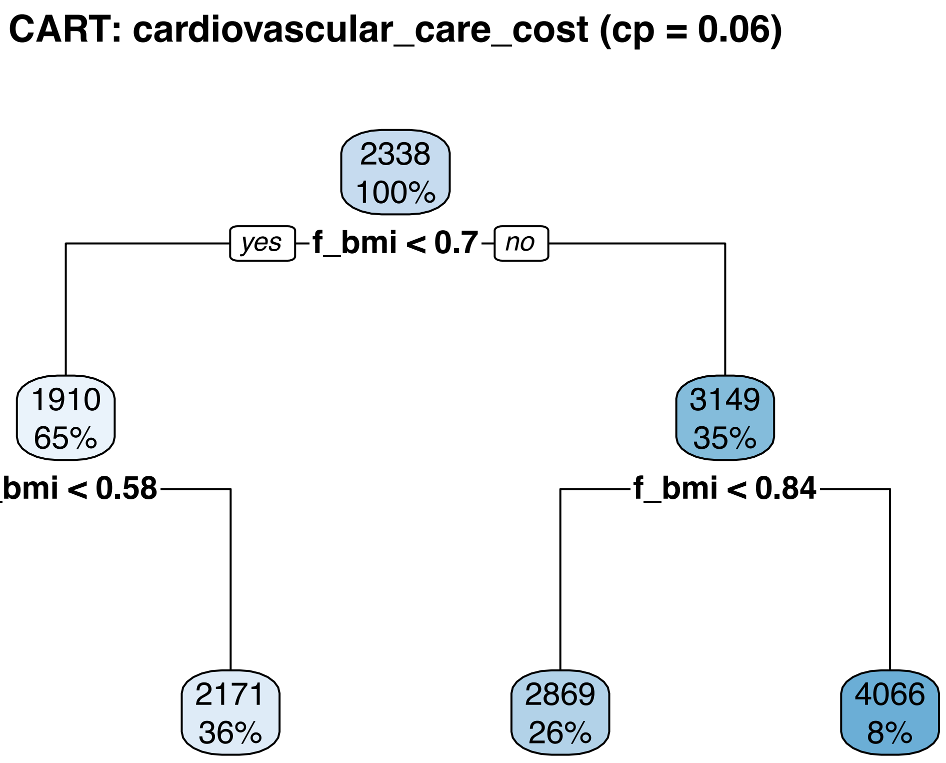
```
##
##
## ## cp = 0.02
```


CART: cardiovascular_care_cost (cp = 0.02)
2338
100%
yes
f_bmi < 0.7
no
1910
65%
3149
35%
f_bmi < 0.58
f_bmi < 0.84
2869
26%
4066
8%
f_bmi < 0.76
f_bmi < 0.95
1593
30%
2171
36%
2650
14%
3103
13%
3796
6%
5062
2%

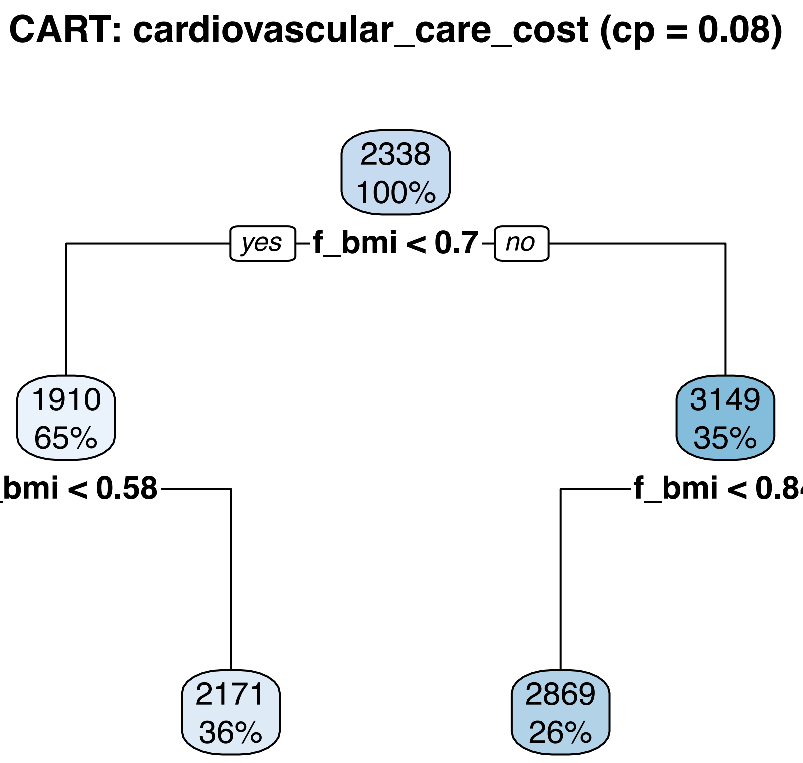
```
##
##
## ## cp = 0.04
```


CART: cardiovascular_care_cost (cp = 0.04)
2338
100%
yes
f_bmi < 0.7
no
1910
65%
f_bmi < 0.58
3149
35%
f_bmi < 0.84
1593
30%
2171
36%
2869
26%
4066
8%

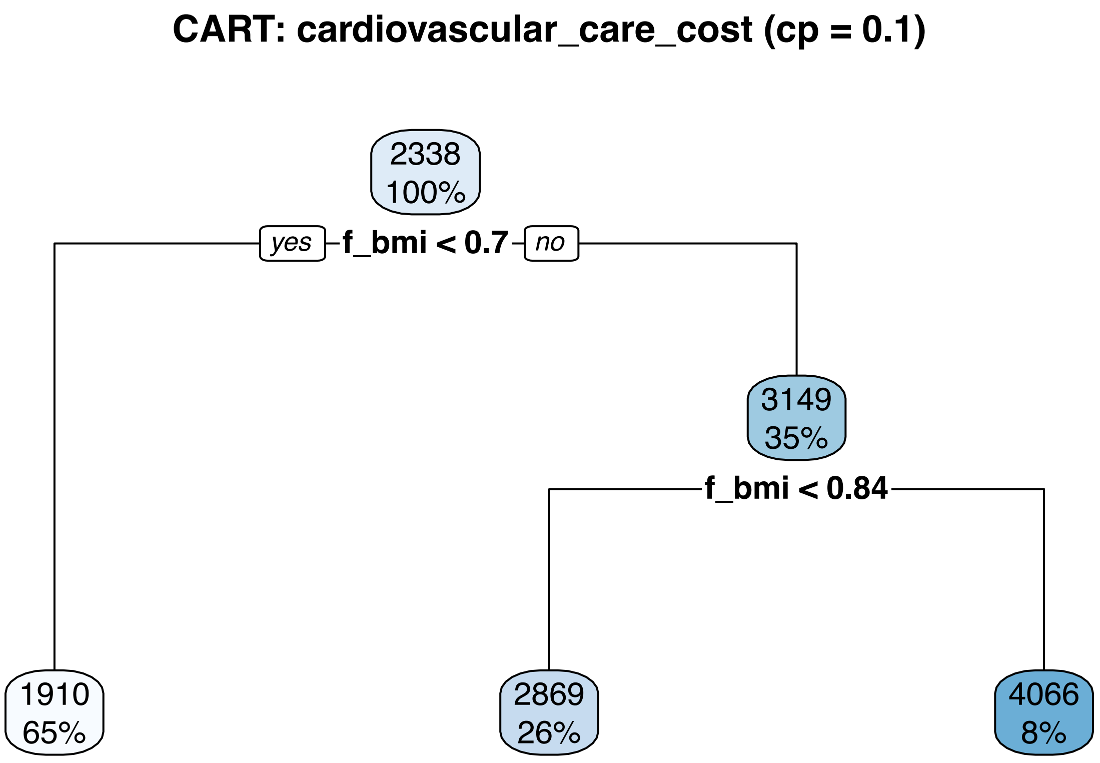
```
##
##
## ## cp = 0.06
```


CART: cardiovascular_care_cost (cp = 0.06)
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 = 0.08
```


CART: cardiovascular_care_cost (cp = 0.08)
2338
100%
yes
f_bmi < 0.7
no
1910
65%
f_bmi < 0.58
3149
35%
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 deeper trees with very high training R^2 (~0.95) but clear overfitting, while larger cp values prune the tree aggressively, reducing variance but increasing bias. The appropriate cp should be chosen by cross-validated error using the 1-SE rule. In this case, cp around 0.04–0.06 yields a shallow tree (4 leaves, depth 2) with R^2 ~0.83 — a balance between interpretability and reasonable fit.

Problem 2

2025-09-24

Problem 2:

```
# NOTE: Data files must be in a 'Data' subdirectory relative to this R Markdown file
# Expected structure:
#   - prob4.Rmd
#   - Data/
#     |- laptop_train.csv
#     |- laptop_test.csv
setwd(dirname(rstudioapi::getSourceEditorContext()$path))

# Read CSV files using relative paths
df_orig <- 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.
```

Question (a)

```
str(df_orig) # structure (variable types, first few entries)
```

```
## spc_tbl_ [1,338 x 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ age                     : num [1:1338] 19 18 28 33 32 31 46 37 37 60 ...
##  $ bmi                     : num [1:1338] 3.9 5.19 5 3.03 4.09 ...
##  $ charges                 : num [1:1338] 16885 1726 4449 21984 3867 ...
##  $ f_bmi                   : num [1:1338] 0.591 0.716 0.699 0.481 0.612 ...
##  $ cardiovascular_care_cost: num [1:1338] 1876 2466 2473 1656 1933 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   bmi = col_double(),
##   ..   charges = col_double(),
##   ..   f_bmi = col_double(),
##   ..   cardiovascular_care_cost = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
summary(df_orig) # summary statistics by column
```

```
##       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
```

```r
# Split the data into training and test (70% train, 30% test)
set.seed(15072)
# training (70%) and test (30%) partition
smp_size <- floor(0.70 * nrow(df_orig))
train_ind <- sample(seq_len(nrow(df_orig)), size = smp_size, replace = FALSE)
df_train <- df_orig[train_ind, ]
df_test <- df_orig[-train_ind, ]

# Check the dimensions of the training and test sets
dim(df_train)
```

```
## [1] 936   5
```

```r
dim(df_test)
```

```
## [1] 402   5
```

```r
# Use glimpse to get a quick overview of both datasets
glimpse(df_train)
```

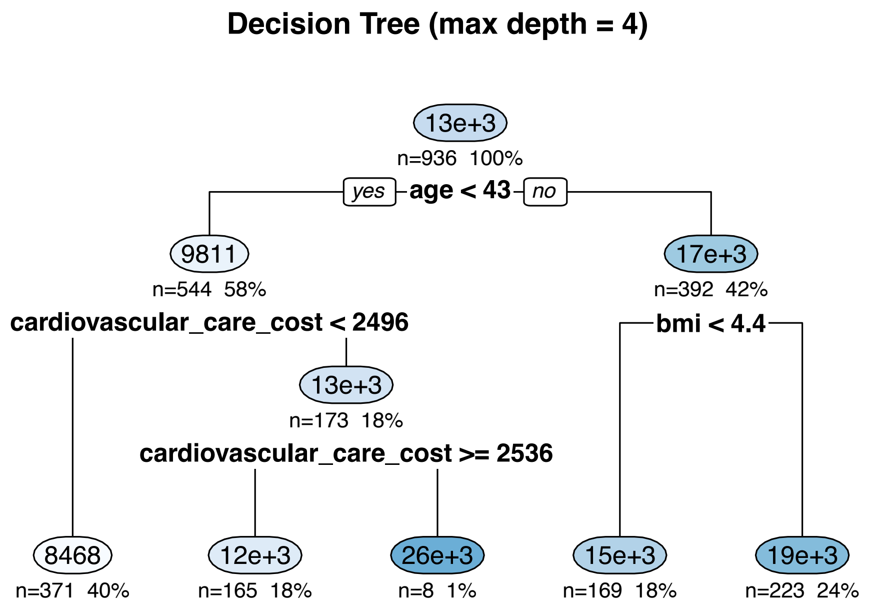
```
## Rows: 936
## Columns: 5
## $ age                      <dbl> 63, 19, 44, 21, 34, 40, 56, 38, 47, 34, 36, 3~
## $ bmi                      <dbl> 5.877215, 3.306357, 3.825654, 3.504007, 3.825~
## $ charges                  <dbl> 13887.204, 2709.112, 7626.993, 17942.106, 500~
## $ f_bmi                    <dbl> 0.7691716, 0.5193497, 0.5827057, 0.5445650, 0~
## $ cardiovascular_care_cost <dbl> 2949.277, 1530.945, 1821.328, 1364.722, 2143.~
```

```r
glimpse(df_test)
```

```
## Rows: 402
## Columns: 5
## $ age                      <dbl> 33, 32, 31, 37, 62, 56, 52, 23, 59, 22, 28, 3~
## $ bmi                      <dbl> 3.027632, 4.092107, 3.510852, 4.286243, 3.606~
## $ charges                  <dbl> 21984.471, 3866.855, 3756.622, 6406.411, 2780~
## $ f_bmi                    <dbl> 0.4811030, 0.6119470, 0.5454126, 0.6320768, 0~
## $ cardiovascular_care_cost <dbl> 1656.104, 1933.298, 1548.085, 1752.809, 1625.~
```

Question (b)

```
# Fit a decision tree model with no depth greater than 4
tree_model <- rpart(charges ~ ., data = df_train, control = rpart.control(maxdepth = 4))


# Plot the decision tree
rpart.plot(tree_model, type = 2, extra = 101, under = TRUE,
           main = "Decision Tree (max depth = 4)")
```



```
# Summary of the model
summary(tree_model)
```

```
## Call:
## rpart(formula = charges ~ ., data = df_train, control = rpart.control(maxdepth = 4))
##   n= 936
##
##           CP nsplit rel error    xerror       xstd
## 1 0.09583610      0 1.0000000 1.0041683 0.06325607
## 2 0.01611526      1 0.9041639 0.9188786 0.05829077
## 3 0.01181389      2 0.8880486 0.9629522 0.05839692
## 4 0.01102792      3 0.8762347 0.9791362 0.05889894
## 5 0.01000000      4 0.8652068 0.9902592 0.05904446
```

```
##
## Variable importance
##                     age cardiovascular_care_cost                      bmi
##                      50                       21                       14
##                   f_bmi
##                      14
##
## Node number 1: 936 observations,    complexity param=0.0958361
##   mean=12914.59, MSE=1.395087e+08
##   left son=2 (544 obs) right son=3 (392 obs)
##   Primary splits:
##       age                      < 42.5      to the left,  improve=0.09583610, (0 missing)
##       bmi                      < 4.392632  to the left,  improve=0.03606740, (0 missing)
##       f_bmi                    < 0.6427244 to the left,  improve=0.03606740, (0 missing)
##       cardiovascular_care_cost < 2493.027  to the left,  improve=0.03167217, (0 missing)
##   Surrogate splits:
##       cardiovascular_care_cost < 3015.68   to the left,  agree=0.597, adj=0.038, (0 split)
##       bmi                      < 5.728587  to the left,  agree=0.593, adj=0.028, (0 split)
##       f_bmi                    < 0.7580472 to the left,  agree=0.593, adj=0.028, (0 split)
##
## Node number 2: 544 observations,    complexity param=0.01611526
##   mean=9810.683, MSE=1.159912e+08
##   left son=4 (371 obs) right son=5 (173 obs)
##   Primary splits:
##       cardiovascular_care_cost < 2496.317  to the left,  improve=0.03334962, (0 missing)
##       bmi                      < 4.357944  to the left,  improve=0.02996210, (0 missing)
##       f_bmi                    < 0.6392812 to the left,  improve=0.02996210, (0 missing)
##       age                      < 28.5      to the left,  improve=0.01657340, (0 missing)
##   Surrogate splits:
##       bmi   < 4.991013  to the left,  agree=0.932, adj=0.786, (0 split)
##       f_bmi < 0.6981874 to the left,  agree=0.932, adj=0.786, (0 split)
##
## Node number 3: 392 observations,    complexity param=0.01181389
##   mean=17222.06, MSE=1.402212e+08
##   left son=6 (169 obs) right son=7 (223 obs)
##   Primary splits:
##       bmi                      < 4.392097  to the left,  improve=0.02806535, (0 missing)
##       f_bmi                    < 0.6426714 to the left,  improve=0.02806535, (0 missing)
##       age                      < 58.5      to the left,  improve=0.02175027, (0 missing)
##       cardiovascular_care_cost < 2163.841  to the left,  improve=0.02065833, (0 missing)
##   Surrogate splits:
##       f_bmi                    < 0.6426714 to the left,  agree=1.000, adj=1.000, (0 split)
##       cardiovascular_care_cost < 2200.411  to the left,  agree=0.926, adj=0.828, (0 split)
##
## Node number 4: 371 observations
##   mean=8467.627, MSE=7.249573e+07
##
## Node number 5: 173 observations,    complexity param=0.01102792
##   mean=12690.88, MSE=1.971037e+08
##   left son=10 (165 obs) right son=11 (8 obs)
##   Primary splits:
##       cardiovascular_care_cost < 2535.65   to the right, improve=0.04223086, (0 missing)
##       bmi                      < 8.080433  to the right, improve=0.02730380, (0 missing)
##       f_bmi                    < 0.9074317 to the right, improve=0.02730380, (0 missing)
##       age                      < 18.5      to the left,  improve=0.01676663, (0 missing)
##
## Node number 6: 169 observations
##   mean=14943.28, MSE=5.366741e+07
##
## Node number 7: 223 observations
##   mean=18949.02, MSE=1.988979e+08
##
## Node number 10: 165 observations
##   mean=12055.6, MSE=1.872335e+08
##
## Node number 11: 8 observations
##   mean=25793.53, MSE=2.206731e+08
```

```r
# Predictions on training and test sets
test_pred <- predict(tree_model, newdata = df_test)

# Calculate R-squared for training and test sets
y_true <- df_test$charges
sse <- sum((y_true - test_pred)^2)
sst <- sum((y_true - mean(y_true))^2)
R2_basetree <- 1 - sse/sst

print(paste("R-squared on test set:", round(R2_basetree, 4)))
```

```
## [1] "R-squared on test set: 0.0557"
```

Question (c)

Residuals

The residuals are the differences between the actual charges in the test dataset and the predicted charges from the basetree model:

$$\text{residual}_i = y_i - \hat{y}_i$$

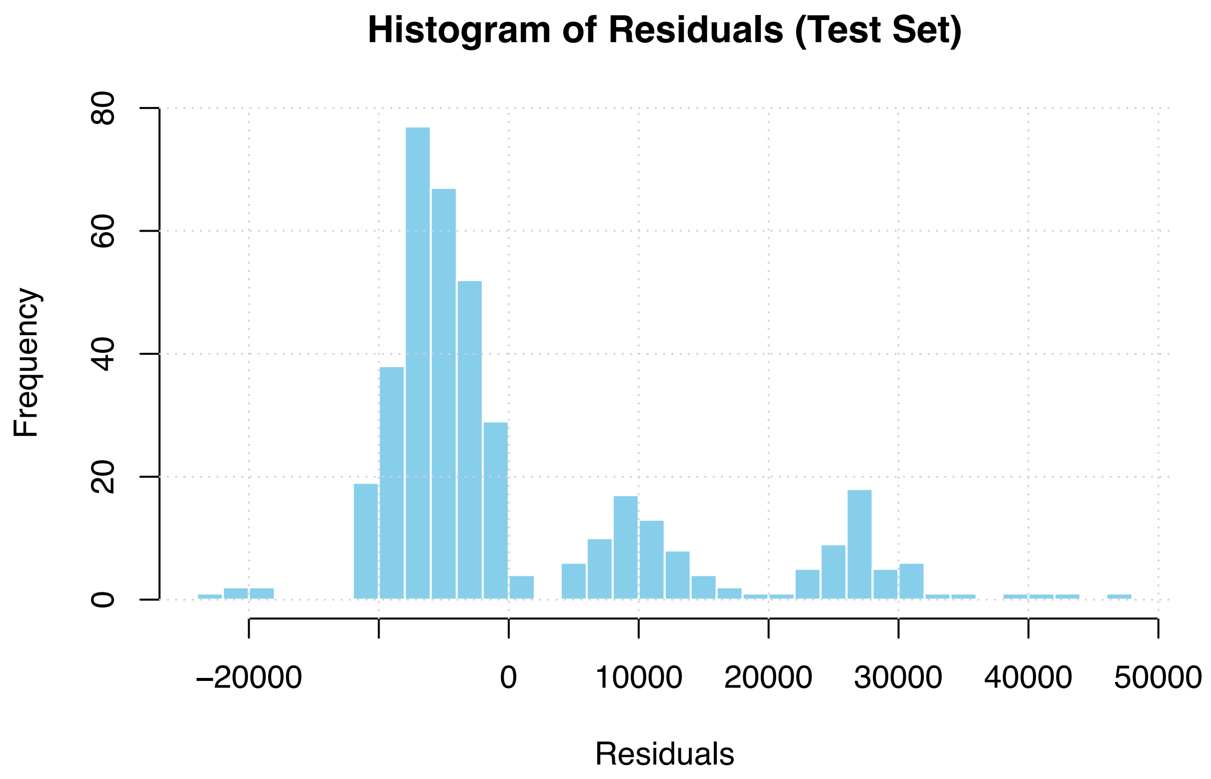
where
- y_i = actual charge for observation i (from the test dataset),
- $\hat{y}_i$ = predicted charge for observation i (from the basetree).

```r
# Calculate residuals on the test set
residuals <- df_test$charges - test_pred

# Summarize the residuals
residual_summary <- summary(residuals)
print(residual_summary)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -23532.0  -6731.6  -4071.9    819.6   7268.4  46515.5
```

We can see that the values of the residuals are quite spread out, indicating that the model has difficulty accurately predicting insurance charges for many individuals. This is also reflected in the R-squared value, which is very low (around 0.0557), indicating that the model explains only a small portion of the variance in the insurance charges.

```r
# Plot a histogram of the residuals
hist(residuals, breaks = 40, main = "Histogram of Residuals (Test Set)",
     xlab = "Residuals", col = "skyblue", border = "white")
grid()
```



Question (d)

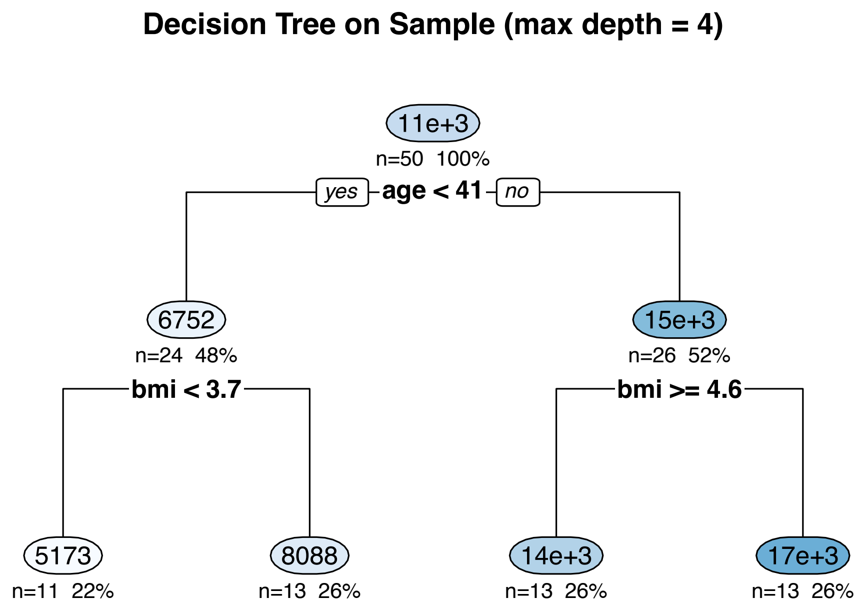
1. First we take a random sample with replacement of size 50 from the training set.

```r
# Take random sample with replacement of size 50 from the training set
df_sample <- df_train[sample(nrow(df_train), 50, replace = TRUE), ]

# Check the dimensions of the sample
dim(df_sample)
```

```
## [1] 50  5
```

2. Next, we fit a decision tree model to this sample, again with max depth 4.

```r
# Fit a decision tree model to the sample
tree_model_sample <- rpart(charges ~ ., data = df_sample,
                           control = rpart.control(maxdepth = 4))

# Plot the decision tree
rpart.plot(tree_model_sample, type = 2, extra = 101, under = TRUE,
           main = "Decision Tree on Sample (max depth = 4)")
```



```r
# Summary of the model
summary(tree_model_sample)
```

```
## Call:
## rpart(formula = charges ~ ., data = df_sample, control = rpart.control(maxdepth = 4))
##   n= 50
##
##           CP nsplit rel error   xerror      xstd
## 1 0.18784602      0 1.0000000 1.023816 0.3615694
## 2 0.01428744      1 0.8121540 0.980413 0.3669908
## 3 0.01060455      2 0.7978665 1.133964 0.3717291
## 4 0.01000000      3 0.7872620 1.144902 0.3722554
##
## Variable importance
##                      age                      bmi                    f_bmi
##                       45                       20                       20
## cardiovascular_care_cost
##                       15
##
## Node number 1: 50 observations,    complexity param=0.187846
##   mean=11159.72, MSE=9.548714e+07
##   left son=2 (24 obs) right son=3 (26 obs)
##   Primary splits:
##       age                      < 40.5      to the left,  improve=0.18784600, (0 missing)
##       bmi                      < 4.223596  to the left,  improve=0.06009507, (0 missing)
##       f_bmi                    < 0.625667  to the left,  improve=0.06009507, (0 missing)
##       cardiovascular_care_cost < 2294.667  to the left,  improve=0.04458673, (0 missing)
##   Surrogate splits:
##       bmi                      < 3.879789  to the left,  agree=0.68, adj=0.333, (0 split)
##       f_bmi                    < 0.5887976 to the left,  agree=0.68, adj=0.333, (0 split)
##       cardiovascular_care_cost < 1984.881  to the left,  agree=0.64, adj=0.250, (0 split)
##
## Node number 2: 24 observations,    complexity param=0.01060455
##   mean=6751.585, MSE=5.709371e+07
##   left son=4 (11 obs) right son=5 (13 obs)
##   Primary splits:
##       bmi                      < 3.701539  to the left,  improve=0.03694943, (0 missing)
##       f_bmi                    < 0.5681381 to the left,  improve=0.03694943, (0 missing)
##       age                      < 28.5      to the right, improve=0.02311741, (0 missing)
##       cardiovascular_care_cost < 2262.969  to the left,  improve=0.02229621, (0 missing)
##   Surrogate splits:
##       f_bmi                    < 0.5681381 to the left,  agree=1.000, adj=1.000, (0 split)
##       cardiovascular_care_cost < 1781.719  to the left,  agree=0.917, adj=0.818, (0 split)
##       age                      < 27        to the left,  agree=0.708, adj=0.364, (0 split)
##
## Node number 3: 26 observations,    complexity param=0.01428744
##   mean=15228.76, MSE=9.643323e+07
##   left son=6 (13 obs) right son=7 (13 obs)
##   Primary splits:
##       bmi                      < 4.621298  to the right, improve=0.02720629, (0 missing)
##       f_bmi                    < 0.6647613 to the right, improve=0.02720629, (0 missing)
##       age                      < 57.5      to the left,  improve=0.02423591, (0 missing)
##       cardiovascular_care_cost < 2294.667  to the left,  improve=0.02419122, (0 missing)
##   Surrogate splits:
##       f_bmi                    < 0.6647613 to the right, agree=1.000, adj=1.000, (0 split)
##       cardiovascular_care_cost < 2070.5    to the right, agree=0.846, adj=0.692, (0 split)
##       age                      < 47.5      to the right, agree=0.615, adj=0.231, (0 split)
##
## Node number 4: 11 observations
##   mean=5172.617, MSE=2.668401e+07
##
## Node number 5: 13 observations
##   mean=8087.635, MSE=7.893039e+07
##
## Node number 6: 13 observations
##   mean=13609.01, MSE=8.389341e+07
##
## Node number 7: 13 observations
##   mean=16848.51, MSE=1.037259e+08
```

3. Finally, we compute the R-squared on the test set using this new model.

```
# Predictions on test set using the model fitted to the sample
test_pred_sample <- predict(tree_model_sample, newdata = df_test)

# Calculate R-squared for test set using the sample model
```

```r
y_true <- df_test$charges
sse_sample <- sum((y_true - test_pred_sample)^2)
sst <- sum((y_true - mean(y_true))^2)
R2_sample <- 1 - sse_sample/sst
print(paste("R-squared on test set (sample model):", round(R2_sample, 4)))
```

```
## [1] "R-squared on test set (sample model): -0.0404"
```

Having a negative R-squared indicates that the model is performing worse than simply predicting the mean of the response variable. This poor performance indicates that this particular tree, trained on a small bootstrap sample of 50 observations, is not capturing the underlying patterns in the data effectively.

Question (e)

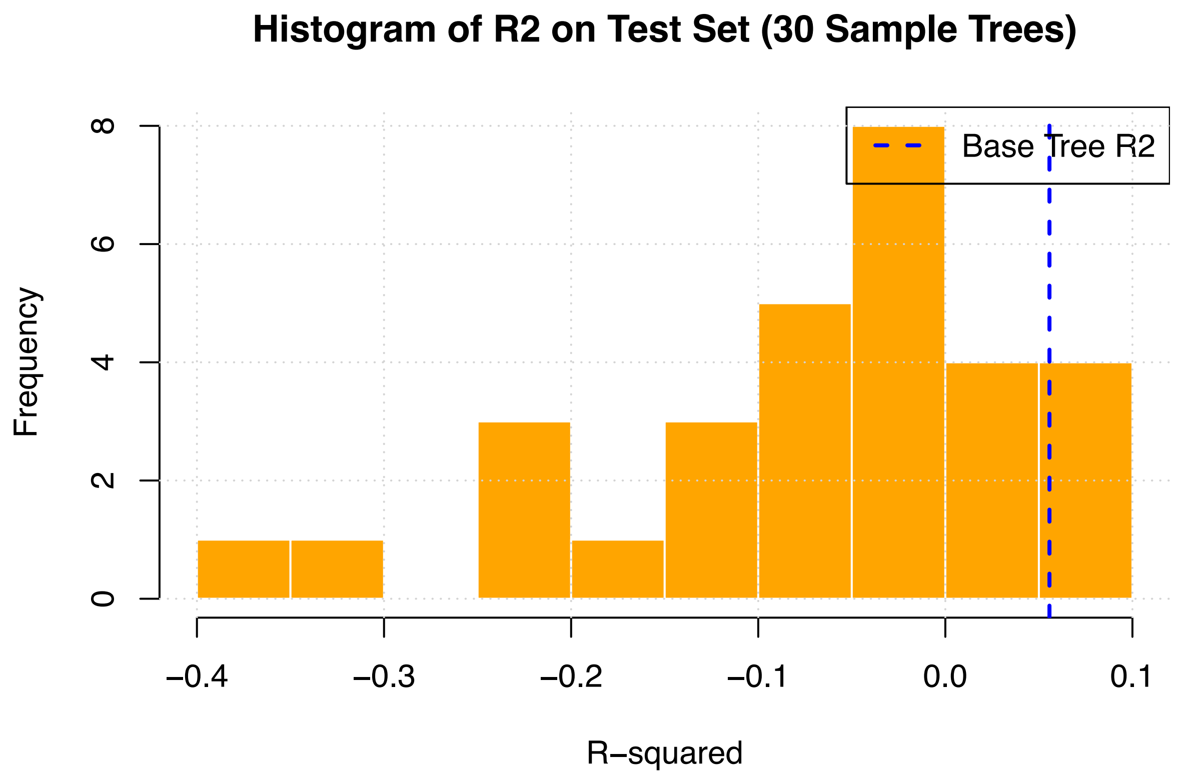
```r
# Repeat the sampling, training, and R2 calculation 30 times
R2_samples <- numeric(30)
wise_tree_models <- vector("list", 30) # Save each tree for use in (f)

for (i in 1:30) {
  # Sample with replacement
  df_sample <- df_train[sample(nrow(df_train), 50, replace = TRUE), ]
  # Fit tree
  tree_model_sample <- rpart(charges ~ ., data = df_sample,
                             control = rpart.control(maxdepth = 4))
  # Store the model
  wise_tree_models[[i]] <- tree_model_sample
  # Predict on test set
  test_pred_sample <- predict(tree_model_sample, newdata = df_test)
  # Compute R2
  y_true <- df_test$charges
  sse_sample <- sum((y_true - test_pred_sample)^2)
  sst <- sum((y_true - mean(y_true))^2)
  R2_samples[i] <- 1 - sse_sample/sst
}

# Print all 30 R2 values, one per line, with index
for (i in 1:30) {
  cat(sprintf("Tree %2d: R-squared on test set = %.4f\n", i, R2_samples[i]))
}
```

```
## Tree  1: R-squared on test set = -0.1308
## Tree  2: R-squared on test set = -0.0180
## Tree  3: R-squared on test set = -0.2456
## Tree  4: R-squared on test set = 0.0144
## Tree  5: R-squared on test set = -0.0797
## Tree  6: R-squared on test set = -0.1790
## Tree  7: R-squared on test set = 0.0681
## Tree  8: R-squared on test set = 0.0597
## Tree  9: R-squared on test set = 0.0632
## Tree 10: R-squared on test set = -0.3430
## Tree 11: R-squared on test set = -0.0013
## Tree 12: R-squared on test set = -0.0490
## Tree 13: R-squared on test set = -0.1315
## Tree 14: R-squared on test set = -0.3888
## Tree 15: R-squared on test set = -0.2100
## Tree 16: R-squared on test set = 0.0092
## Tree 17: R-squared on test set = 0.0037
## Tree 18: R-squared on test set = -0.0150
## Tree 19: R-squared on test set = -0.0479
## Tree 20: R-squared on test set = -0.0585
## Tree 21: R-squared on test set = -0.0672
## Tree 22: R-squared on test set = -0.0909
## Tree 23: R-squared on test set = -0.0463
## Tree 24: R-squared on test set = -0.0486
## Tree 25: R-squared on test set = -0.2147
## Tree 26: R-squared on test set = -0.1230
## Tree 27: R-squared on test set = 0.0071
## Tree 28: R-squared on test set = 0.0701
## Tree 29: R-squared on test set = -0.0299
## Tree 30: R-squared on test set = -0.0682
```

```r
# Print basetree R2 for comparison
cat(sprintf("\nBasetree: R-squared on test set = %.4f\n", R2_basetree))
```

```
##
## Basetree: R-squared on test set = 0.0557
```

```r
# Plot histogram of the 30 R2 values
hist(R2_samples, breaks = 10, main = "Histogram of R2 on Test Set (30 Sample Trees)",
     xlab = "R-squared", col = "orange", border = "white")
abline(v = R2_basetree, col = "blue", lwd = 2, lty = 2)
legend("topright", legend = "Base Tree R2", col = "blue", lwd = 2, lty = 2)
grid()
```



Results

Across the 30 bootstrapped trees (each trained on a random sample of 50 points with replacement, max depth = 4), the test-set R^2 values ranged from about -0.39 to +0.07, with most values negative. This means the bootstrapped trees generally performed worse than predicting the mean of the response. In contrast, the basetree (trained on the full training data with the same depth limit) achieved a small but positive R^2 of 0.0557, indicating that access to the full dataset led to slightly better generalization. A histogram of the 30 values would show them tightly clustered around negative performance, while the basetree stands just above zero.

The poor results arise mainly from two factors.

1. First, using only 50 observations in each bootstrap sample provides too little information, producing highly variable and weak models. Because sampling is done with replacement, some observations appear multiple times while others are skipped, reducing even further the variety of data the model sees.
2. Second, limiting tree depth to 4 constrains model complexity, so the trees cannot capture important structure in the data.

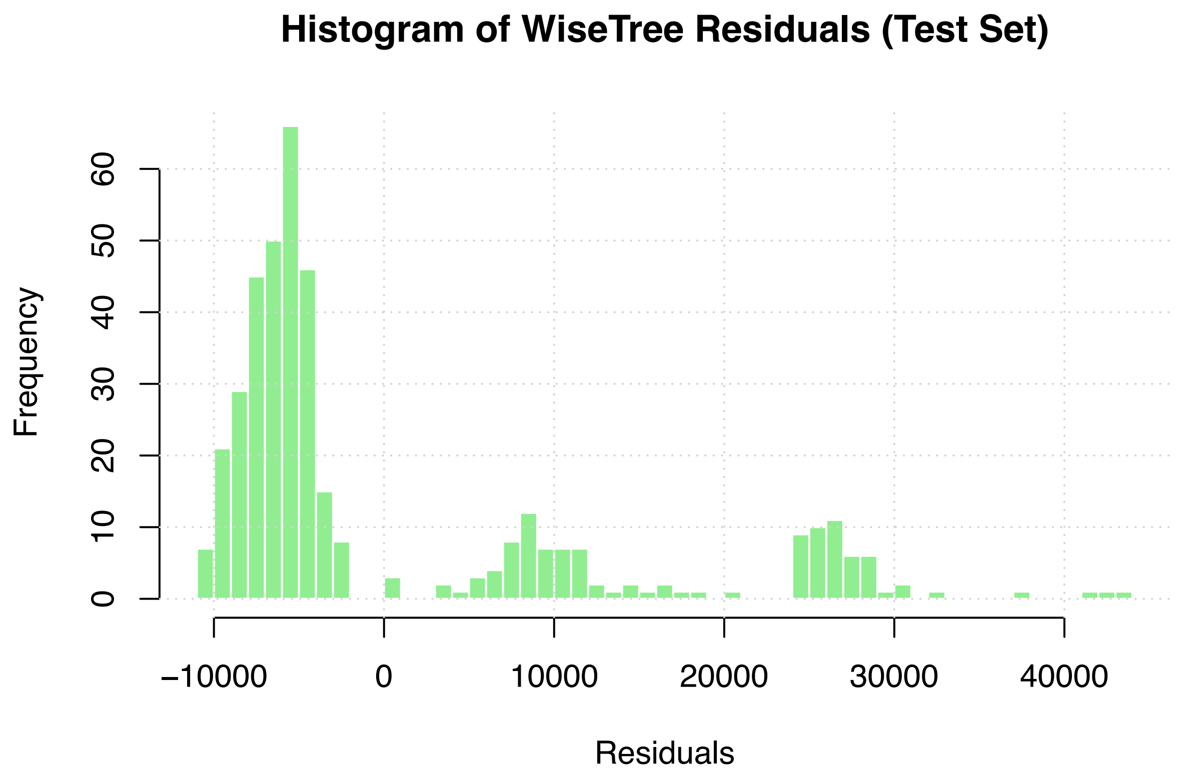
Question (f)

Now we are going to create `WiseTree`, an ensemble of 30 trees, each trained on a different bootstrap sample of size 50 (with replacement) from the training set, and each with max depth 4. In this ensemble, predictions are made by averaging the predictions of all 30 trees.

```r
# Use the 30 pretrained trees from wise_tree_models

# Make predictions by averaging the predictions of all trees (WiseTree ensemble)
wise_tree_pred <- rowMeans(sapply(wise_tree_models, function(model)
                                  predict(model, newdata = df_test)))

# Calculate residuals for WiseTree
wise_tree_residuals <- df_test$charges - wise_tree_pred

# Summarize the residuals
wise_tree_residuals_summary <- summary(wise_tree_residuals)
print(wise_tree_residuals_summary)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10977.0  -7022.2  -5260.9    435.2   7110.7  43943.3
```

```r
# Plot histogram of the residuals
hist(wise_tree_residuals, breaks = 40, main = "Histogram of WiseTree Residuals (Test Set)",
     xlab = "Residuals", col = "lightgreen", border = "white")
grid()
```



Here we also see that the residuals are quite spread out, indicating that the ensemble model still has difficulty accurately predicting insurance charges for many individuals. However, the spread of the residuals appears to be slightly less extreme compared to the single trees (question (c)), suggesting that averaging predictions from multiple trees helps to stabilize the predictions somewhat.

Question (g)

```r
# Calculate R-squared for WiseTree on the test set
y_true <- df_test$charges
sse_wisetree <- sum((y_true - wise_tree_pred)^2)
sst <- sum((y_true - mean(y_true))^2)
R2_wisetree <- 1 - sse_wisetree/sst
cat(sprintf("WiseTree: R-squared on test set = %.4f\n", R2_wisetree))
```

```
## WiseTree: R-squared on test set = 0.0950
```

```r
# Compare with basetree R2
cat(sprintf("Basetree: R-squared on test set = %.4f\n", R2_basetree))
```

```
## Basetree: R-squared on test set = 0.0557
```

WiseTree outperforms the basetree because it averages predictions from 30 bootstrapped trees, giving it more stability and better generalization. Bootstrapping samples the training set with replacement, so each tree sees a slightly different dataset, introducing diversity among the models. Averaging across these diverse but shallow trees reduces variance, making WiseTree more reliable than a single tree trained on the full dataset.

Problem 3

2025-09-24

Problem 3: Logistic Regression Redux

```
# NOTE: Data files must be in a 'Data' subdirectory relative to this R Markdown file
# Expected structure:
#   - prob4.Rmd
#   - Data/
#     |- laptop_train.csv
#     |- laptop_test.csv
setwd(dirname(rstudioapi::getSourceEditorContext()$path))

# Read CSV files using relative paths
df_orig <- read_csv("../customerchurn.csv")
```

```
## Rows: 7032 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): PaymentMethod, InternetService, Contract
## dbl (4): Churn, MonthlyCharges, SeniorCitizen, tenure
##
## 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.
```

Question (a)

```
str(df_orig) # structure (variable types, first few entries)
```

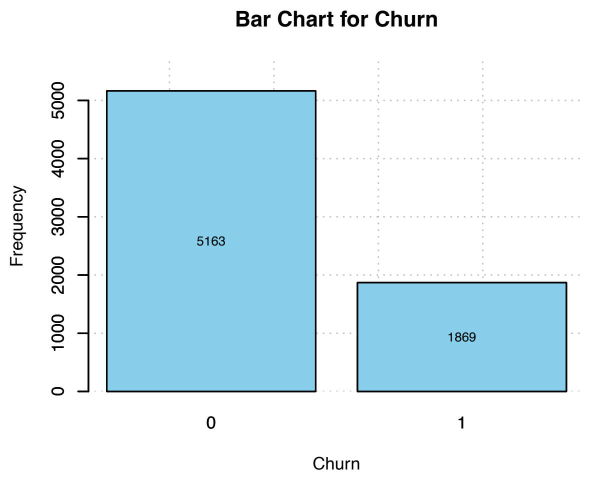
```
## spc_tbl_ [7,032 x 7] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Churn          : num [1:7032] 0 0 1 0 1 1 0 0 1 0 ...
##  $ MonthlyCharges : num [1:7032] 29.9 57 53.9 42.3 70.7 ...
##  $ SeniorCitizen  : num [1:7032] 0 0 0 0 0 0 0 0 0 0 ...
##  $ PaymentMethod  : chr [1:7032] "Electronic check" "Mailed check" "Mailed check" "Bank transfer" ..
##  $ InternetService: chr [1:7032] "DSL" "DSL" "DSL" "DSL" ...
##  $ tenure         : num [1:7032] 1 34 2 45 2 8 22 10 28 62 ...
##  $ Contract       : chr [1:7032] "Month-to-month" "One year" "Month-to-month" "One year" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Churn = col_double(),
##   ..   MonthlyCharges = col_double(),
##   ..   SeniorCitizen = col_double(),
##   ..   PaymentMethod = col_character(),
##   ..   InternetService = col_character(),
##   ..   tenure = col_double(),
##   ..   Contract = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

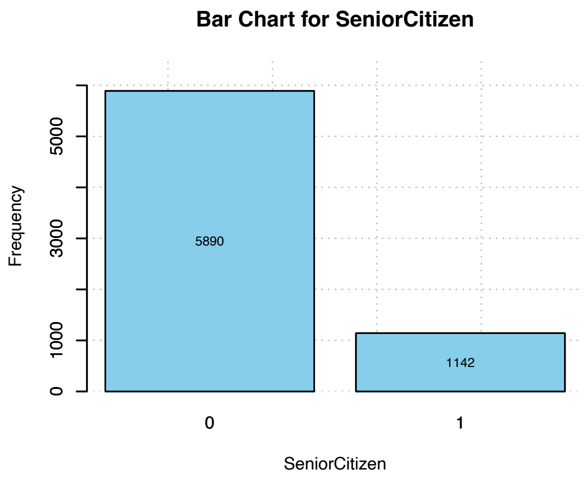
```
summary(df_orig) # summary statistics by column
```

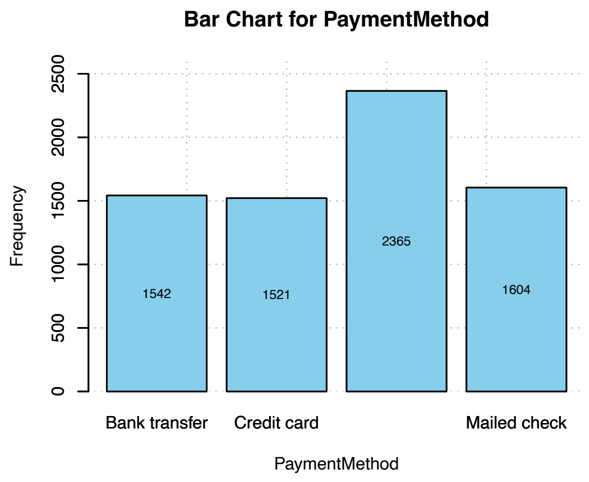
```
##      Churn         MonthlyCharges   SeniorCitizen    PaymentMethod
##  Min.   :0.0000   Min.   : 18.25   Min.   :0.0000   Length:7032
##  1st Qu.:0.0000   1st Qu.: 35.59   1st Qu.:0.0000   Class :character
##  Median :0.0000   Median : 70.35   Median :0.0000   Mode  :character
##  Mean   :0.2658   Mean   : 64.80   Mean   :0.1624
##  3rd Qu.:1.0000   3rd Qu.: 89.86   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :118.75   Max.   :1.0000
##  InternetService        tenure        Contract
##  Length:7032        Min.   : 1.00   Length:7032
##  Class :character   1st Qu.: 9.00   Class :character
##  Mode  :character   Median :29.00   Mode  :character
##                     Mean   :32.42
##                     3rd Qu.:55.00
##                     Max.   :72.00
```

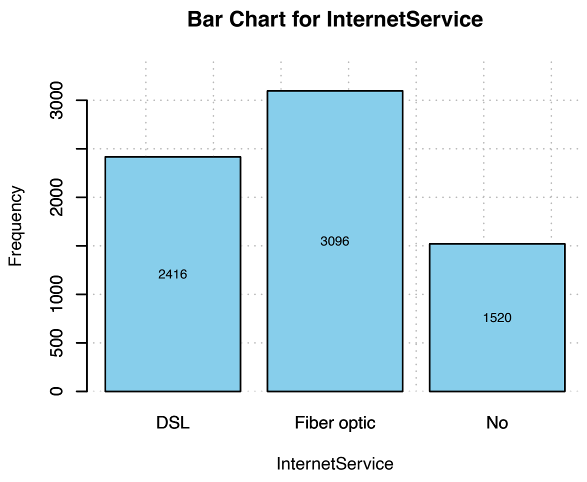
```
# Set up 3x2 grid layout
par(mfrow = c(3, 2))

barchartvector = c("Churn", "SeniorCitizen",
                    "PaymentMethod", "InternetService", "Contract")
for (col_name in barchartvector) {
  counts <- table(df_orig[[col_name]])

  # Create the barplot and store the bar positions
  bar_positions <- barplot(counts,
          main = paste("Bar Chart for", col_name),
          xlab = col_name,
          ylab = "Frequency",
          col = "skyblue",
          ylim = c(0, max(counts) * 1.1))  # Add some space at top

  # Add grid
  grid(nx = NULL, ny = NULL, col = "gray", lty = "dotted", lwd = 1)

  # Redraw the bars on top of the grid
  barplot(counts,
          col = "skyblue",
          add = TRUE)

  # Add text labels inside each bar
  text(x = bar_positions,
       y = counts / 2,  # Position at middle of each bar
       labels = counts,
       cex = 0.8,  # Text size
```

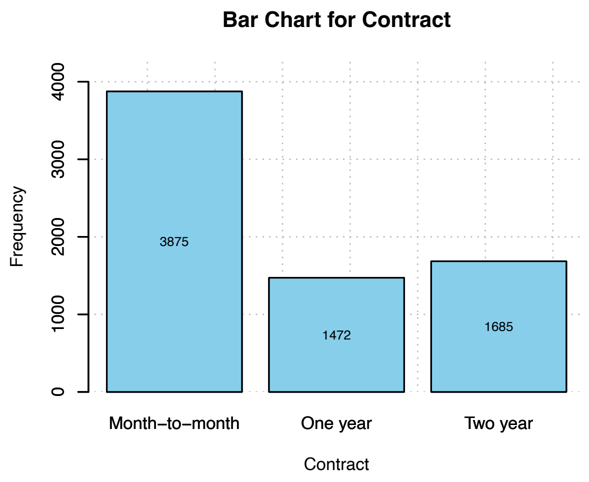
```
        col = "black")   # Text color for visibility
}
```



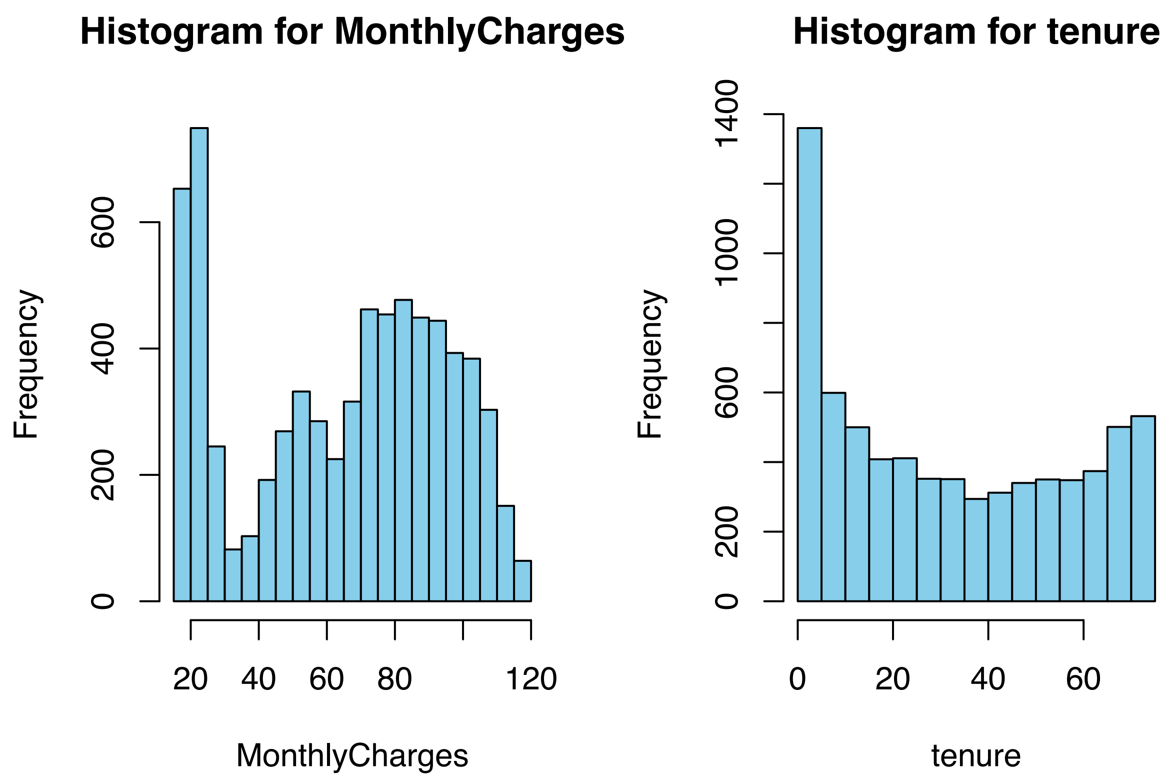








```r
# Set up 1x2 grid layout for histograms
par(mfrow = c(1, 2))

histvector = c("MonthlyCharges", "tenure")
for (col_name in histvector) {
  # Use the actual column values, not table counts
  hist(df_orig[[col_name]],
          main = paste("Histogram for", col_name),
          xlab = col_name,
          ylab = "Frequency",
          col = "skyblue", breaks = "FD")
}
```



```r
# Reset to default single plot layout
par(mfrow = c(1, 1))
```

From this output, we have a few key takeaways: the churn percentage is 26.5785% and the non churn percentage is 73.4215%, meaning we have approx 3x more observations of non churners than churners. We also realize that our data set is also unbalanced in terms of age demographics, having only 16.24% senior citizens. For the payment methods, users have the choice between 4 different payment methods including electronic check which is the most popular choice, and bank transfer/credit card/mailed check which all have pretty similar utilization. To provide more color on internet service, we see that the most popular service is fiber optic followed by DSL, but many people report not having a service. Moreover, the majority of people pay for these services month to month rather than being locked in for 1 or 2 years. With these services, we see that users pay on average of \$64.80 per month, with a high of \$118.75 and a low of \$18.25. Charges are skewed right as is the histogram of tenure.

```r
# Use table() function to see the distribution of the target variable
churn_table <- table(df_orig$Churn)
print(churn_table)
```

```
##
##    0    1
## 5163 1869
```

```r
# Number of customers who did not churn (Churn = 0)
customers_not_churned <- churn_table[1]
cat("Customers who did not churn:", customers_not_churned, "\n")
```

```
## Customers who did not churn: 5163
```

```r
# Number of customers who churned (Churn = 1)
customers_churned <- churn_table[2]
cat("Customers who churned:", customers_churned, "\n")
```

```
## Customers who churned: 1869
```

```r
# Compute churn rate (proportion of customers who churned)
churn_rate <- customers_churned / sum(churn_table)
cat("Churn rate:", round(churn_rate, 4), "or", round(churn_rate * 100, 2), "%\n")
```

```
## Churn rate: 0.2658 or 26.58 %
```

In the above we used table function on the variable Churn to see some characteristics of the variable. We can see that the variable is binary with 0 and 1 values.

Analysis of Customer Churn

From the `table()` function results, we can answer the key questions:

How many customers churned? - 1,869 customers churned (Churn = 1)

How many customers did not churn? - 5,163 customers did not churn (Churn = 0)

What is the customers' churn rate? - The churn rate is **26.58%** (or 0.2658 as a proportion) - This means that approximately 1 out of every 4 customers in the dataset churned

Summary: - Total customers: 7,032 (5,163 + 1,869) - Non-churn rate: 73.42% - Churn rate: 26.58%

This indicates a relatively high churn rate, as over a quarter of the customer base has churned. This level of churn represents a significant business concern that warrants further analysis to identify the factors contributing to customer attrition.

[Optional] We can also compute the churn rate using different methods as shown below.

```r
# To compute customer's churn rate
# Method 1: Using mean() function (churn is doubled encoded as 0/1)
churn_rate <- mean(df_orig$Churn == 1)
print(churn_rate)
```

```
## [1] 0.265785
```

Question (b)

```r
set.seed(15072)
# creating a binary dependent variable for churn (0 = no, 1 = churn)
df_orig$Churn <- as.factor(df_orig$Churn)

# training (70%) and test (30%) partition
smp_size <- floor(0.70 * nrow(df_orig))
train_ind <- sample(seq_len(nrow(df_orig)), size = smp_size, replace = FALSE)
df_train <- df_orig[train_ind, ]
df_test <- df_orig[-train_ind, ]
```

```r
# Fit logistic regression model on training data
mod_logit <- glm(Churn ~ . - PaymentMethod, data = df_train, family = binomial)
summary(mod_logit)
```

```
##
## Call:
## glm(formula = Churn ~ . - PaymentMethod, family = binomial, data = df_train)
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -0.492422   0.188257  -2.616   0.0089 **
## MonthlyCharges               0.003523   0.003632   0.970   0.3321
## SeniorCitizen                0.460054   0.095973   4.794 1.64e-06 ***
## InternetServiceFiber optic  1.041354   0.153243   6.795 1.08e-11 ***
## InternetServiceNo           -0.886428   0.177866  -4.984 6.24e-07 ***
## tenure                      -0.033687   0.002497 -13.492  < 2e-16 ***
## ContractOne year            -0.829613   0.124337  -6.672 2.52e-11 ***
## ContractTwo year            -1.736190   0.212439  -8.173 3.02e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5763.5  on 4921  degrees of freedom
## Residual deviance: 4227.2  on 4914  degrees of freedom
## AIC: 4243.2
##
## Number of Fisher Scoring iterations: 6
```

```r
# State the value of and interpret the coefficient for SeniorCitizen
coef_summary <- summary(mod_logit)$coefficients
senior_coef <- coef_summary["SeniorCitizen", "Estimate"]
cat(sprintf("Coefficient for SeniorCitizen: %.4f\n", senior_coef))
```

```
## Coefficient for SeniorCitizen: 0.4601
```

Interpretation:

In the logistic regression model, the `SeniorCitizen` coefficient represents the change in the log-odds of churn for senior citizens compared to non-senior citizens, holding all other variables constant. Since the coefficient is positive (0.4601), it means that being a senior citizen increases the likelihood of customer churn. Specifically, the odds of churn for senior citizens are exp(0.4601) ≈ 1.58 times the odds for non-senior citizens, all else being the same. In other words, senior citizens are about 58% more likely to churn than non-senior citizens, holding all other factors constant.

Question (c)

So essentially I need to find the 5th customer from the initial dataset and predict the probability of churn for that customer using the fitted logistic regression model.

1. Get the 5th customer from the original dataset, keeping only the features used to fit the model

```
customer_5 <- df_orig[5, ]
print(customer_5)
```

```
## # A tibble: 1 x 7
##   Churn MonthlyCharges SeniorCitizen PaymentMethod    InternetService tenure
##   <fct>          <dbl>         <dbl> <chr>            <chr>            <dbl>
## 1 1               70.7             0 Electronic check Fiber optic          2
## # i 1 more variable: Contract <chr>
```

2. Predict the probability of churn for the 5th customer

```
churn_prob <- predict(mod_logit, newdata = customer_5, type = "response")
cat(sprintf("Predicted probability of churn for customer 5: %.4f\n", churn_prob))
```

```
## Predicted probability of churn for customer 5: 0.6749
```

Question (d)

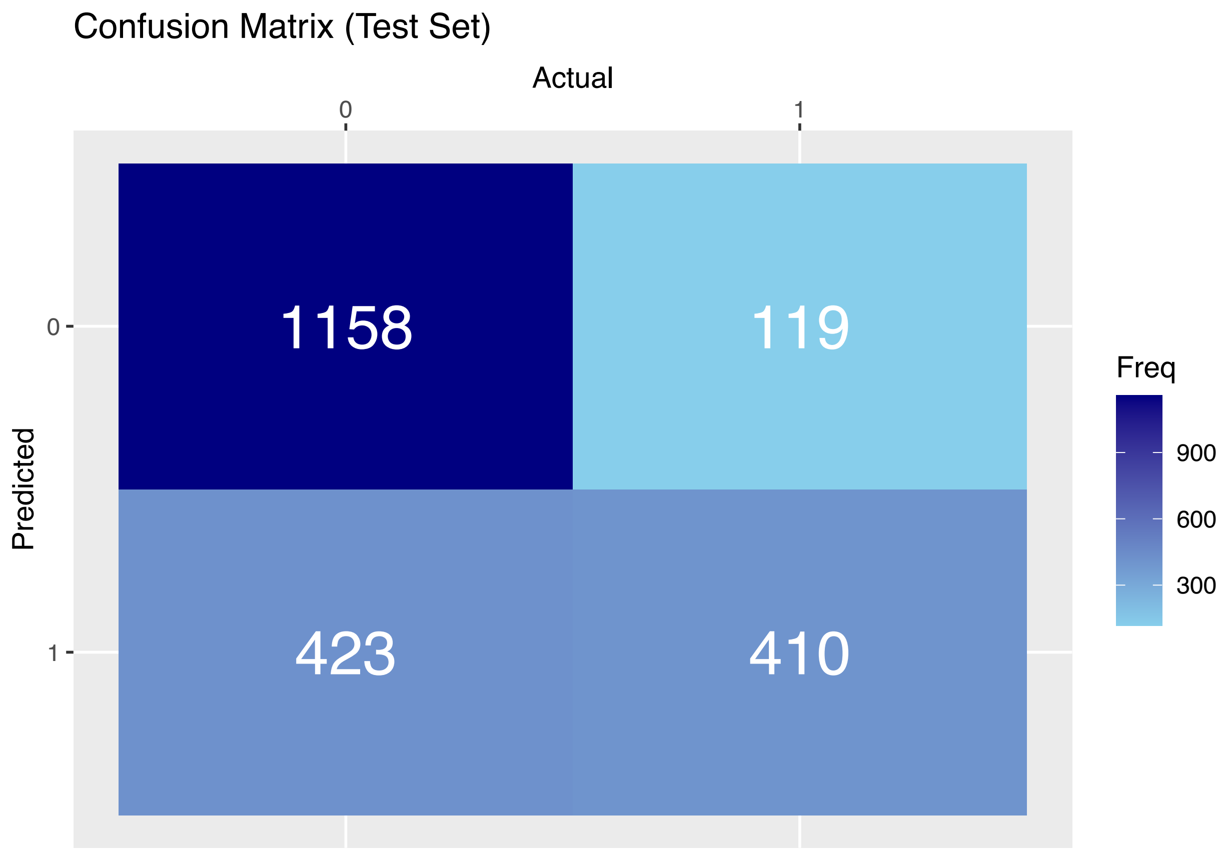
1. Predict probabilities on the test set

```
# Predict probabilities on test set
pred_probs <- predict(mod_logit, newdata = df_test, type = "response")

# Classify based on threshold of 0.3 (any probability higher
# than 30% will be considered as churn)
pred_classes <- ifelse(pred_probs > 0.3, 1, 0)

# Confusion matrix
conf_matrix <- table(Predicted = pred_classes, Actual = df_test$Churn)
print(conf_matrix)
```

```
##          Actual
## Predicted    0    1
##         0 1158  119
##         1  423  410
```

2. Visualize confusion matrix

```r
# Visualize confusion matrix as heatmap
cm_df <- as.data.frame(conf_matrix)
colnames(cm_df) <- c("Predicted", "Actual", "Freq")
cm_df$Predicted <- factor(cm_df$Predicted, levels = c(1, 0))
cm_df$Actual <- factor(cm_df$Actual, levels = c(0, 1))

ggplot(cm_df, aes(x = Actual, y = Predicted, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), color = "white", size = 8) +
  scale_fill_gradient(low = "skyblue", high = "navy") +
  labs(
    title = "Confusion Matrix (Test Set)",
    x = "Actual",
    y = "Predicted"
  ) +
  scale_x_discrete(position = "top") # Put Actual labels on top to match table
```



Question (e)

In the confusion matrix, we can identify the following:

- False Positives (FP): These are the cases where the model predicted churn (1) but the actual outcome was no churn (0). In the confusion matrix, this is represented by the cell where Predicted = 1 and Actual = 0. From the confusion matrix, we can see that there are 422 false positives.

- False Negatives (FN): These are the cases where the model predicted no churn (0) but the actual outcome was churn (1). In the confusion matrix, this is represented by the cell where Predicted = 0 and Actual = 1. From the confusion matrix, we can see that there are 119 false negatives.

Question (f)

Here we need to intepret the tradeoff between false positives and false negatives in a managerial context. First of all we need to understand that from a company's perspective, failing to identify a customer who will churn is often more costly than incorrectly predicting that a customer will churn when they won't. This is because if you can identify a customer who is likely to churn, you can take proactive measures to retain them, resulting in a direct financial benefit. On the other hand, if you incorrectly predict that a customer will churn, the cost is usually limited to the resources spent on retention efforts, which may not be as significant as losing a customer.

Now let's analyze the tradeoff from a data perspective: Based on the above, we can understand that if we need to decide between minimizing false positives or false negatives, we should prioritize minimizing false negatives (meaning wrongly predicting a customer will not churn when they actually do).

Therefore, from a managerial perspective, minimizing false negatives (meaning wrongly predicting a customer will not churn when they actually do) is more critical, because missing a true churner means losing revenue and potentially long-term customer value. While false positives incur costs (e.g., retention offers to loyal customers), these costs are generally smaller and can even strengthen customer relationships. As an analyst, I would prioritize a model that does a better job of catching as many real churners as possible, even if it means we sometimes intervene with customers who would have stayed anyway.

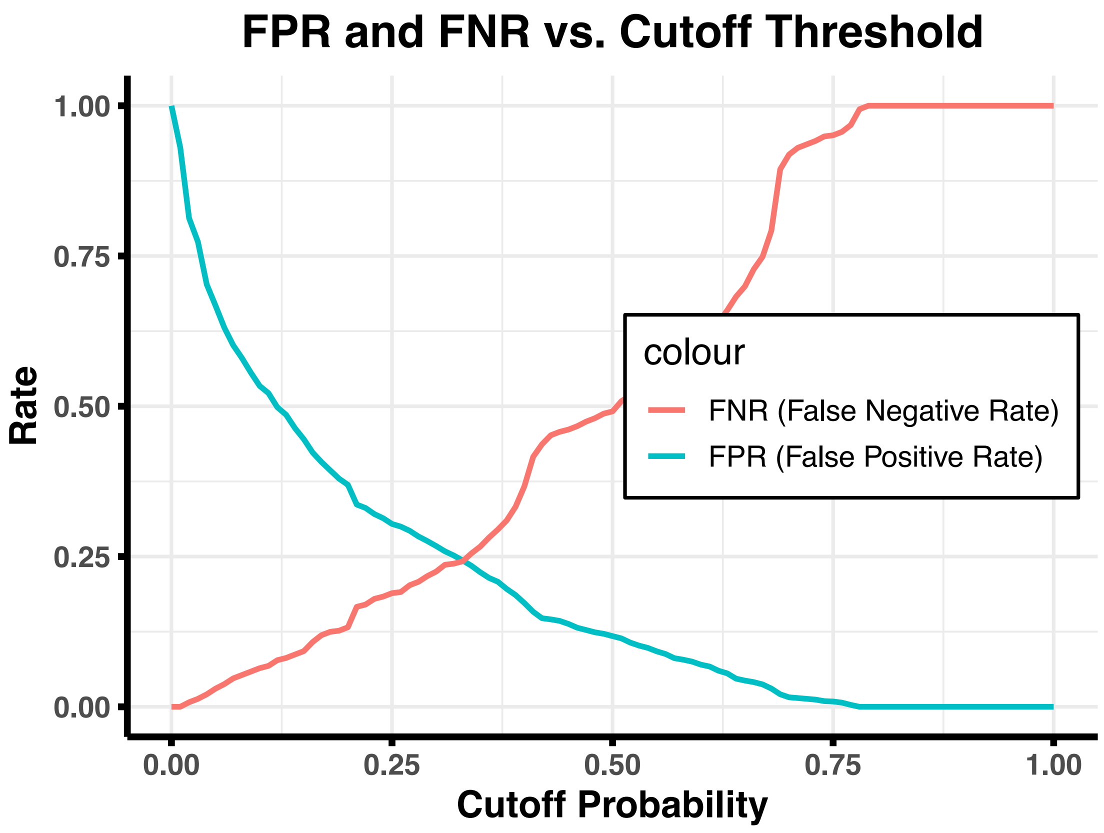
Question (g)

```r
# Threshold plot: FPR and FNR vs cutoff

cutoffs <- seq(0, 1, by = 0.01)
fpr <- fnr <- numeric(length(cutoffs))

actual <- df_test$Churn

for (i in seq_along(cutoffs)) {
  pred <- ifelse(pred_probs > cutoffs[i], 1, 0)
  cm <- table(factor(pred, levels = c(0,1)), factor(actual, levels = c(0,1)))
  # cm[1,1]=TN, cm[1,2]=FN, cm[2,1]=FP, cm[2,2]=TP
  fp <- cm[2,1]
  fn <- cm[1,2]
  tn <- cm[1,1]
  tp <- cm[2,2]
  fpr[i] <- ifelse((fp + tn) > 0, fp / (fp + tn), NA)
  fnr[i] <- ifelse((fn + tp) > 0, fn / (fn + tp), NA)
}

threshold_df <- data.frame(
  Cutoff = cutoffs,
  FPR = fpr,
  FNR = fnr
)

ggplot(threshold_df, aes(x = Cutoff)) +
  geom_line(aes(y = FPR, color = "FPR (False Positive Rate)"), size = 1) +
  geom_line(aes(y = FNR, color = "FNR (False Negative Rate)"), size = 1) +
  labs(
    title = "FPR and FNR vs. Cutoff Threshold",
    x = "Cutoff Probability",
    y = "Rate"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    axis.title = element_text(face = "bold"),
    axis.text = element_text(face = "bold"),
    plot.title = element_text(face = "bold", hjust = 0.5),
    axis.line = element_line(size = 1.2, colour = "black"),
    axis.ticks = element_line(size = 1.2, colour = "black"),
    plot.margin = margin(10, 30, 10, 30),
    legend.position = c(0.98, 0.5),           # right end, middle
    legend.justification = c("right", "center"),
    legend.background = element_rect(fill = "white", color = "black")
  ) +
  coord_cartesian(xlim = c(0, 1), ylim = c(0, 1))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
## i Please use the `linewidth` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
## 3.5.0.
## i Please use the `legend.position.inside` argument of `theme()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



1. In the plot, the red curve represents the false negative rate (FNR) and the blue curve represents the false positive rate (FPR). As the cutoff probability increases along the x-axis, the blue FPR curve steadily declines, showing that fewer loyal customers are misclassified as churners. At the same time, the red FNR curve rises, showing that more true churners are missed by the model. Both curves are plotted on the same scale, so the crossing point in the middle range makes it easy to see where the balance between the two errors occurs. At the extremes, the tradeoff is clear: at threshold 0 FPR = 1 and FNR = 0 (everyone is flagged as churner), while at threshold 1 FPR = 0 and FNR = 1 (no churners are caught).
2. This visualization makes the tradeoff very clear: increasing the cutoff reduces false positives but increases false negatives. At very low thresholds, the company would waste significant resources by intervening with many customers who would not leave, while at very high thresholds the company would lose real churners and the revenue tied to them. The most effective managerial decision is to choose a threshold in the middle range, where the balance between the two error rates maximizes profitability and aligns best with business goals.

Question (h)

From a managerial perspective, our goal is to set a probability threshold that maximizes overall profit while balancing the trade-off between false positives and false negatives. The profit matrix makes it clear that missing a customer who churns is extremely costly, whereas wrongly predicting churn for a loyal customer has a smaller financial impact. Therefore, we expect the optimal threshold to lean toward being more aggressive in flagging churners, ensuring that fewer at-risk customers are missed. To identify this point, I evaluated total profit across twenty different thresholds.

The given profit matrix is as follows:

	Actual Churn	Actual Non-Churn
Predicted Churn	**$2000**	**-$1000**
Predicted Non-Churn	**-$6000**	**$3000**

We are plotting the expected profit against 20 different probability thresholds ranging from 0 to 1.

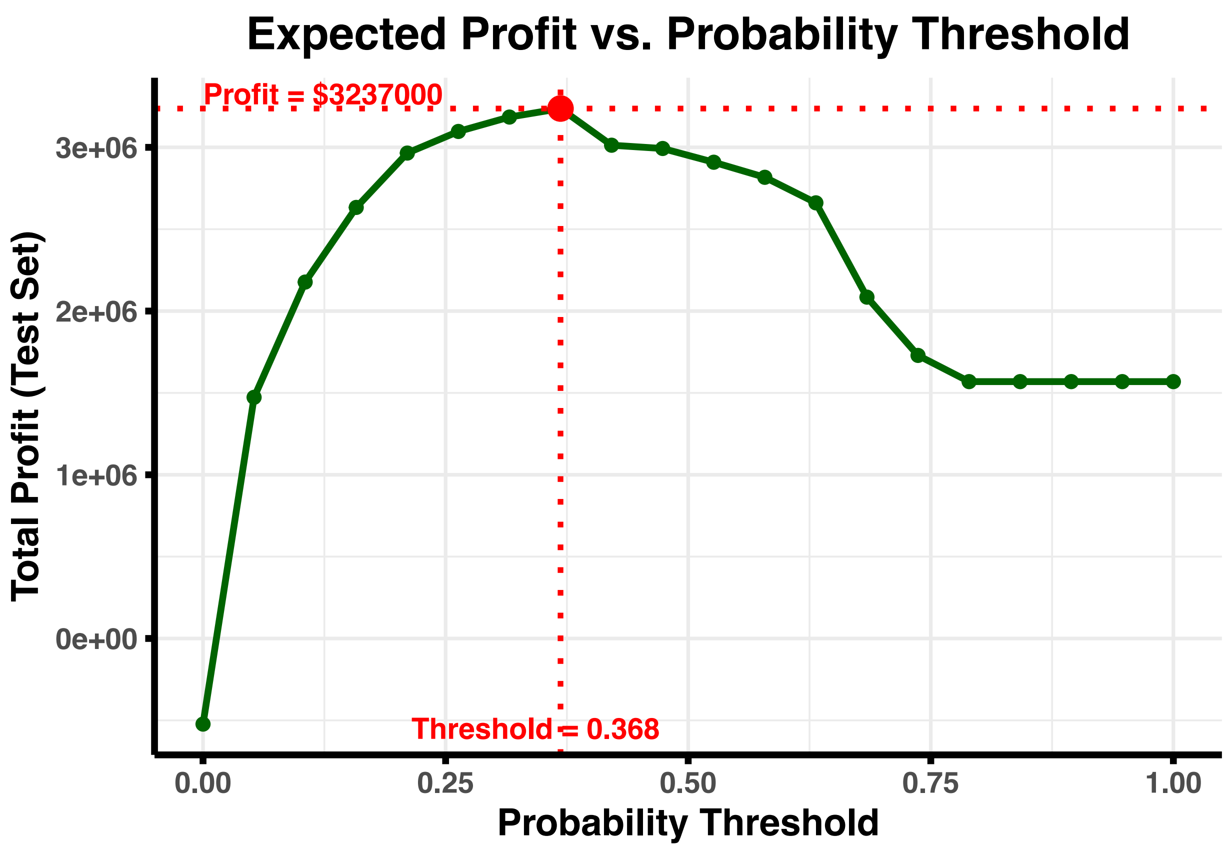
```r
# Define profit matrix
profit_matrix <- matrix(
  c(2000, -1000, -6000, 3000), # order: TP, FP, FN, TN
  nrow = 2, byrow = TRUE,
  dimnames = list(
    "Predicted" = c("Churn", "NonChurn"),
    "Actual" = c("Churn", "NonChurn")
  )
)

# Thresholds: 20 equally spaced values between 0 and 1
thresholds <- seq(0, 1, length.out = 20)
profits <- numeric(length(thresholds))
TPs <- FPs <- FNs <- TNs <- numeric(length(thresholds))

actual <- df_test$Churn

for (i in seq_along(thresholds)) {
  pred <- ifelse(pred_probs > thresholds[i], 1, 0)
  # Confusion matrix: rows = predicted, cols = actual
  cm <- table(factor(pred, levels = c(1,0)), factor(actual, levels = c(1,0)))
  # cm[1,1]=TP, cm[1,2]=FP, cm[2,1]=FN, cm[2,2]=TN
  TP <- cm[1,1]
  FP <- cm[1,2]
  FN <- cm[2,1]
  TN <- cm[2,2]
  profits[i] <- TP*2000 + FP*(-1000) + FN*(-6000) + TN*3000
  TPs[i] <- TP
  FPs[i] <- FP
  FNs[i] <- FN
  TNs[i] <- TN
  cat(sprintf("Threshold: %.3f | TP: %d | FP: %d | TN: %d | FN: %d | Profit: $%d\n",
              thresholds[i], TP, FP, TN, FN, profits[i]))
}
```

```
## Threshold: 0.000 | TP: 529 | FP: 1581 | TN: 0 | FN: 0 | Profit: $-523000
## Threshold: 0.053 | TP: 511 | FP: 1046 | TN: 535 | FN: 18 | Profit: $1473000
## Threshold: 0.105 | TP: 493 | FP: 834 | TN: 747 | FN: 36 | Profit: $2177000
## Threshold: 0.158 | TP: 473 | FP: 680 | TN: 901 | FN: 56 | Profit: $2633000
## Threshold: 0.211 | TP: 440 | FP: 531 | TN: 1050 | FN: 89 | Profit: $2965000
## Threshold: 0.263 | TP: 426 | FP: 470 | TN: 1111 | FN: 103 | Profit: $3097000
## Threshold: 0.316 | TP: 404 | FP: 404 | TN: 1177 | FN: 125 | Profit: $3185000
## Threshold: 0.368 | TP: 374 | FP: 331 | TN: 1250 | FN: 155 | Profit: $3237000
## Threshold: 0.421 | TP: 297 | FP: 233 | TN: 1348 | FN: 232 | Profit: $3013000
## Threshold: 0.474 | TP: 277 | FP: 198 | TN: 1383 | FN: 252 | Profit: $2993000
## Threshold: 0.526 | TP: 249 | FP: 163 | TN: 1418 | FN: 280 | Profit: $2909000
## Threshold: 0.579 | TP: 218 | FP: 124 | TN: 1457 | FN: 311 | Profit: $2817000
## Threshold: 0.632 | TP: 180 | FP: 87 | TN: 1494 | FN: 349 | Profit: $2661000
## Threshold: 0.684 | TP: 83 | FP: 37 | TN: 1544 | FN: 446 | Profit: $2085000
## Threshold: 0.737 | TP: 29 | FP: 18 | TN: 1563 | FN: 500 | Profit: $1729000
## Threshold: 0.789 | TP: 0 | FP: 0 | TN: 1581 | FN: 529 | Profit: $1569000
## Threshold: 0.842 | TP: 0 | FP: 0 | TN: 1581 | FN: 529 | Profit: $1569000
## Threshold: 0.895 | TP: 0 | FP: 0 | TN: 1581 | FN: 529 | Profit: $1569000
## Threshold: 0.947 | TP: 0 | FP: 0 | TN: 1581 | FN: 529 | Profit: $1569000
## Threshold: 1.000 | TP: 0 | FP: 0 | TN: 1581 | FN: 529 | Profit: $1569000
```

```r
# Create data frame for plotting
profit_df <- data.frame(
  Threshold = thresholds,
  Profit = profits
)

# Print the best threshold and its stats
best_idx <- which.max(profit_df$Profit)
cat(sprintf("\nBest Threshold: %.3f | TP: %d | FP: %d | TN: %d | FN: %d | Max Profit: $%d\n",
            profit_df$Threshold[best_idx], TPs[best_idx], FPs[best_idx],
            TNs[best_idx], FNs[best_idx], profit_df$Profit[best_idx]))
```

```
##
## Best Threshold: 0.368 | TP: 374 | FP: 331 | TN: 1250 | FN: 155 | Max Profit: $3237000
```

```r
ggplot(profit_df, aes(x = Threshold, y = Profit)) +
  geom_line(color = "darkgreen", size = 1.2) +
  geom_point(color = "darkgreen", size = 2) +
  # Highlight best threshold with a red circle
  geom_point(
    data = profit_df[best_idx, , drop = FALSE],
    aes(x = Threshold, y = Profit),
    color = "red", size = 4, shape = 21, fill = "red"
  ) +
  # Add dotted lines to axes from the best point
  geom_vline(
    xintercept = profit_df$Threshold[best_idx],
    linetype = "dotted", color = "red", size = 1
  ) +
  geom_hline(
    yintercept = profit_df$Profit[best_idx],
    linetype = "dotted", color = "red", size = 1
  ) +
  # Annotate the best point with threshold and profit values
  annotate(
    "text",
    x = profit_df$Threshold[best_idx],
    y = min(profit_df$Profit),
    label = sprintf("Threshold = %.3f", profit_df$Threshold[best_idx]),
    vjust = 0.7, hjust = 0.6, color = "red", fontface = "bold"
  ) +
  annotate(
    "text",
    x = min(profit_df$Threshold),
    y = profit_df$Profit[best_idx],
    label = sprintf("Profit = $%d", profit_df$Profit[best_idx]),
    vjust = -0.2, hjust = 0, color = "red", fontface = "bold"
  ) +
  labs(
    title = "Expected Profit vs. Probability Threshold",
    x = "Probability Threshold",
    y = "Total Profit (Test Set)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    axis.title = element_text(face = "bold"),
    axis.text = element_text(face = "bold"),
    axis.line = element_line(size = 1.2, colour = "black"),
    axis.ticks = element_line(size = 1.2, colour = "black"),
    plot.title = element_text(face = "bold", hjust = 0.5)
  )
```



The results confirm this intuition. At very low thresholds, the model predicts nearly everyone as a churner, leading to heavy losses from excessive false positives. As the threshold increases, profits rise steadily, reaching a maximum of $3,237,000 at a threshold of 0.368. At this point, the model correctly identifies 373 churners, with 331 false positives and 155 false negatives. Beyond this threshold, profits start to decline, as the number of missed churners grows and outweighs the savings from fewer false positives.

In the plot, I highlighted this optimal threshold with red dotted lines. It clearly represents the point where the trade-off between catching churners and avoiding unnecessary retention costs delivers the highest return. From a business standpoint, this suggests adopting a moderately low threshold that focuses on reducing costly missed churners while still managing retention expenses.

R setup

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(fitdistrplus)
```

```
## Loading required package: MASS
```

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

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

```
## Loading required package: survival
```

```
library(ggplot2)
library(olsrr)
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:MASS':
##
##     cement
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

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

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:survival':
##
##     cluster
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

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

```r
library(glue)
```

Problem 4

Setup

```
set.seed(15072)

df_ames <- read_csv("/Users/riyaparikh_computeracct/Downloads/MIT/15.072_AdvancedAnalyti
csEdge/deliverable2-analyticsedge-mit/ames.csv", show_col_types = FALSE)

# training (70%) and test (30%) partition
nrow = nrow(df_ames)
train_index <- sample(1:nrow, size = 0.7* nrow)
train_ames <- df_ames[train_index, ]
test_ames <- df_ames[-train_index, ]
```

Part a

```
set.seed(15072)
mod_linear_intial <- lm(SalePrice ~ ., data = train_ames)
summary(mod_linear_intial)
```

```
##
## Call:
## lm(formula = SalePrice ~ ., data = train_ames)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -304836  -10699     -42   10472  129099
##
## Coefficients: (7 not defined because of singularities)
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -3.177e+05  8.923e+05  -0.356 0.721876
## MSZoningRL               4.069e+03  3.950e+03   1.030 0.303126
## MSZoningRM               4.903e+03  4.666e+03   1.051 0.293482
## LotFrontage             -1.302e+02  4.290e+01  -3.035 0.002437 **
## LotArea                  1.527e-01  1.702e-01   0.897 0.369689
## StreetPave              -9.429e+02  1.511e+04  -0.062 0.950249
## AlleyNo Alley            4.545e+03  3.329e+03   1.365 0.172329
## AlleyPave                2.907e+03  5.056e+03   0.575 0.565398
## LotShapeMod+ IR         -1.345e+03  3.536e+03  -0.380 0.703705
## LotShapeReg             -6.858e+02  1.367e+03  -0.502 0.616051
## LandContourHLS           1.889e+04  4.199e+03   4.498 7.31e-06 ***
## LandContourLow           4.018e+03  5.384e+03   0.746 0.455535
## LandContourLvl           1.167e+04  3.084e+03   3.782 0.000161 ***
## LotConfigCulDSac         6.310e+02  3.035e+03   0.208 0.835326
## LotConfigFR2            -5.913e+03  3.624e+03  -1.632 0.102931
## LotConfigFR3            -4.961e+02  7.475e+03  -0.066 0.947086
## LotConfigInside         -1.746e+03  1.552e+03  -1.125 0.260796
## LandSlopeNot Gtl         9.969e+03  3.339e+03   2.986 0.002866 **
## NeighborhoodBlueste     -7.305e+03  1.301e+04  -0.561 0.574572
## NeighborhoodBrDale       9.959e+03  9.714e+03   1.025 0.305392
## NeighborhoodBrkSide     -6.574e+03  8.148e+03  -0.807 0.419900
## NeighborhoodClearCr      7.643e+03  8.260e+03   0.925 0.354913
## NeighborhoodCollgCr      2.730e+03  6.628e+03   0.412 0.680465
## NeighborhoodCrawfor      2.022e+04  7.442e+03   2.717 0.006654 **
## NeighborhoodEdwards     -1.947e+04  7.089e+03  -2.746 0.006086 **
## NeighborhoodGilbert     -6.953e+03  6.904e+03  -1.007 0.313999
## NeighborhoodGreens       1.098e+04  1.394e+04   0.788 0.430975
## NeighborhoodIDOTRR      -1.297e+04  8.623e+03  -1.504 0.132676
## NeighborhoodMeadowV     -1.733e+04  8.862e+03  -1.955 0.050728 .
## NeighborhoodMitchel     -7.251e+03  7.151e+03  -1.014 0.310751
## NeighborhoodNAmes       -8.649e+03  7.014e+03  -1.233 0.217709
## NeighborhoodNoRidge      3.915e+04  7.527e+03   5.201 2.21e-07 ***
## NeighborhoodNPkVill      1.368e+04  9.349e+03   1.464 0.143501
## NeighborhoodNridgHt      4.249e+04  6.880e+03   6.175 8.15e-10 ***
## NeighborhoodNWAmes      -4.111e+03  7.192e+03  -0.572 0.567597
## NeighborhoodOldTown     -1.447e+04  8.170e+03  -1.771 0.076773 .
## NeighborhoodSawyer      -8.414e+03  7.225e+03  -1.165 0.244366
## NeighborhoodSawyerW     -1.783e+03  6.879e+03  -0.259 0.795571
## NeighborhoodSomerst      2.453e+04  7.379e+03   3.325 0.000903 ***
## NeighborhoodStoneBr      4.084e+04  7.629e+03   5.353 9.79e-08 ***
## NeighborhoodSWISU       -1.278e+04  8.391e+03  -1.523 0.127936
## NeighborhoodTimber       1.118e+04  7.199e+03   1.553 0.120521
## NeighborhoodVeenker      3.282e+04  9.467e+03   3.467 0.000539 ***
## Condition1Feedr          2.851e+03  4.172e+03   0.683 0.494457
## Condition1Norm           1.107e+04  3.486e+03   3.176 0.001521 **
## Condition1PosA           2.523e+04  7.255e+03   3.478 0.000518 ***
## Condition1PosN           1.302e+04  5.864e+03   2.221 0.026490 *
## Condition1RRAe          -7.330e+02  6.691e+03  -0.110 0.912777
## Condition1RRAn           3.548e+03  5.709e+03   0.621 0.534376
## Condition1RRNe          -3.459e+03  1.142e+04  -0.303 0.762068
## Condition1RRNn           7.658e+03  1.165e+04   0.657 0.510981
## Condition2Other         -9.149e+03  5.933e+03  -1.542 0.123246
## BldgType2fmCon          -4.286e+03  4.744e+03  -0.904 0.366333
## BldgTypeDuplex          -9.778e+03  5.290e+03  -1.848 0.064702 .
## BldgTypeTwnhs           -3.281e+04  4.805e+03  -6.829 1.17e-11 ***
## BldgTypeTwnhsE          -2.633e+04  3.242e+03  -8.120 8.56e-16 ***
## HouseStyle1.5Unf         2.420e+03  7.672e+03   0.315 0.752518
## HouseStyle1Story         2.689e+02  3.201e+03   0.084 0.933060
## HouseStyle2.5Fin        -6.680e+03  1.322e+04  -0.505 0.613509
## HouseStyle2.5Unf        -8.216e+02  7.093e+03  -0.116 0.907809
## HouseStyle2Story        -2.559e+02  2.709e+03  -0.094 0.924765
## HouseStyleSFoyer        -7.635e+02  4.698e+03  -0.163 0.870919
## HouseStyleSLvl          -3.142e+03  4.035e+03  -0.779 0.436318
## YearBuilt                1.014e+02  6.223e+01   1.630 0.103226
## YearRemodAdd             2.293e+02  4.299e+01   5.335 1.08e-07 ***
## RoofStyleGable           6.241e+03  1.026e+04   0.608 0.542992
## RoofStyleGambrel         5.263e+03  1.210e+04   0.435 0.663666
## RoofStyleHip             9.114e+03  1.029e+04   0.885 0.376122
## RoofStyleMansard         1.070e+04  1.463e+04   0.731 0.464798
## RoofStyleShed            2.681e+04  1.498e+04   1.790 0.073599 .
## RoofMatlOther            2.234e+03  6.935e+03   0.322 0.747375
## Exterior1stMetalSd      -3.430e+03  7.028e+03  -0.488 0.625576
## Exterior1stOther         7.347e+03  3.580e+03   2.052 0.040284 *
## Exterior1stVinylSd      -1.209e+04  7.327e+03  -1.650 0.099182 .
## Exterior1stWd Sdng      -2.111e+03  4.512e+03  -0.468 0.639957
## Exterior2ndMetalSd       8.355e+03  7.041e+03   1.187 0.235517
## Exterior2ndOther        -3.036e+03  3.545e+03  -0.856 0.391840
## Exterior2ndVinylSd       1.525e+04  7.359e+03   2.072 0.038427 *
## Exterior2ndWd Sdng       6.332e+03  4.589e+03   1.380 0.167758
## MasVnrTypeBrkFace        5.037e+03  6.667e+03   0.755 0.450049
## MasVnrTypeNone           5.767e+03  6.673e+03   0.864 0.387601
## MasVnrTypeStone          3.461e+03  6.933e+03   0.499 0.617665
## MasVnrArea               4.298e+00  5.231e+00   0.822 0.411420
## ExterQualFa             -1.902e+04  8.283e+03  -2.296 0.021803 *
## ExterQualGd             -1.110e+04  4.087e+03  -2.715 0.006681 **
## ExterQualTA             -2.006e+04  4.527e+03  -4.430 9.98e-06 ***
## ExterCondFa             -1.396e+04  1.118e+04  -1.248 0.212233
## ExterCondGd             -4.832e+03  1.011e+04  -0.478 0.632622
## ExterCondTA             -4.194e+03  1.006e+04  -0.417 0.676711
## FoundationCBlock         4.958e+03  2.542e+03   1.950 0.051307 .
## FoundationPConc          5.283e+03  2.828e+03   1.868 0.061925 .
## FoundationSlab           6.251e+03  7.628e+03   0.819 0.412638
## FoundationStone          3.633e+03  9.178e+03   0.396 0.692304
## FoundationWood           7.703e+02  1.254e+04   0.061 0.951014
## BsmtQualFa              -1.975e+04  4.962e+03  -3.981 7.14e-05 ***
## BsmtQualGd              -1.771e+04  2.817e+03  -6.286 4.07e-10 ***
## BsmtQualNo Basement     -2.763e+04  9.904e+03  -2.790 0.005326 **
## BsmtQualTA              -1.560e+04  3.503e+03  -4.454 8.93e-06 ***
## BsmtCondGd               3.513e+03  4.220e+03   0.832 0.405304
## BsmtCondNo Basement             NA         NA      NA       NA
## BsmtCondPo              -3.550e+02  1.783e+04  -0.020 0.984114
## BsmtCondTA               1.987e+03  3.266e+03   0.608 0.542935
## BsmtExposureGd           1.072e+04  2.555e+03   4.193 2.89e-05 ***
## BsmtExposureMn          -6.323e+03  2.564e+03  -2.466 0.013746 *
## BsmtExposureNo Basement         NA         NA      NA       NA
## BsmtExposureNo Exposure -4.749e+03  1.907e+03  -2.490 0.012858 *
## BsmtFinType1BLQ         -6.945e+02  2.341e+03  -0.297 0.766775
## BsmtFinType1GLQ          5.324e+03  2.119e+03   2.512 0.012088 *
## BsmtFinType1LwQ         -6.536e+03  2.935e+03  -2.227 0.026072 *
## BsmtFinType1No Basement         NA         NA      NA       NA
## BsmtFinType1Rec         -3.041e+03  2.407e+03  -1.263 0.206743
## BsmtFinType1Unf         -3.161e+03  2.423e+03  -1.304 0.192289
## BsmtFinSF1               1.083e+01  3.943e+00   2.748 0.006057 **
## BsmtFinType2BLQ         -2.744e+03  5.329e+03  -0.515 0.606691
## BsmtFinType2GLQ          3.599e+03  6.266e+03   0.574 0.565824
## BsmtFinType2LwQ         -4.531e+03  5.176e+03  -0.875 0.381506
## BsmtFinType2No Basement         NA         NA      NA       NA
## BsmtFinType2Rec         -2.265e+03  4.986e+03  -0.454 0.649663
## BsmtFinType2Unf          7.536e+02  5.028e+03   0.150 0.880878
## BsmtFinSF2               1.401e+01  6.868e+00   2.039 0.041547 *
## BsmtUnfSF                8.281e+00  3.673e+00   2.255 0.024268 *
## TotalBsmtSF                     NA         NA      NA       NA
## HeatingHotW              3.684e+03  6.128e+03   0.601 0.547788
## HeatingOther            -5.718e+02  9.070e+03  -0.063 0.949734
## HeatingQCFa             -7.305e+03  3.544e+03  -2.061 0.039442 *
## HeatingQCGd             -1.609e+03  1.704e+03  -0.944 0.345282
## HeatingQCTA             -4.376e+03  1.683e+03  -2.599 0.009413 **
## CentralAirY              1.661e+03  3.011e+03   0.552 0.581256
## ElectricalFF            -3.990e+01  4.859e+03  -0.008 0.993449
## ElectricalFP             7.554e+03  1.145e+04   0.660 0.509590
## ElectricalSB             3.966e+01  2.449e+03   0.016 0.987081
## X1stFlrSF                4.387e+01  4.207e+00  10.429  < 2e-16 ***
## X2ndFlrSF                4.113e+01  4.771e+00   8.621  < 2e-16 ***
## LowQualFinSF             2.020e+01  1.331e+01   1.517 0.129328
## GrLivArea                       NA         NA      NA       NA
## BsmtFullBath             6.939e+03  1.581e+03   4.389 1.21e-05 ***
## BsmtHalfBath            -1.128e+03  2.367e+03  -0.476 0.633849
## FullBath                 4.710e+03  1.821e+03   2.586 0.009783 **
## HalfBath                -7.456e+02  1.693e+03  -0.440 0.659707
## BedroomAbvGr             4.360e+02  1.103e+03   0.395 0.692567
## KitchenAbvGr            -1.196e+04  4.621e+03  -2.589 0.009693 **
## KitchenQualFa           -2.634e+04  5.315e+03  -4.955 7.92e-07 ***
## KitchenQualGd           -2.299e+04  3.207e+03  -7.170 1.09e-12 ***
## KitchenQualTA           -2.741e+04  3.550e+03  -7.721 1.91e-14 ***
## TotRmsAbvGrd            -2.736e+02  7.780e+02  -0.352 0.725121
## FunctionalMaj2          -2.363e+04  1.270e+04  -1.861 0.062964 .
## FunctionalMin1          4.314e+03  8.237e+03   0.524 0.600515    
## FunctionalMin2          2.858e+02  8.181e+03   0.035 0.972138    
## FunctionalMod          -3.856e+02  8.921e+03  -0.043 0.965529    
## FunctionalTyp           1.336e+04  7.400e+03   1.805 0.071181 .  
## Fireplaces              8.233e+03  2.242e+03   3.673 0.000247 ***
## FireplaceQuFa          -2.211e+04  5.698e+03  -3.881 0.000108 ***
## FireplaceQuGd          -1.648e+04  4.547e+03  -3.624 0.000298 ***
## FireplaceQuNo Fireplace -1.560e+04  5.286e+03  -2.952 0.003195 ** 
## FireplaceQuPo          -1.267e+04  6.688e+03  -1.894 0.058365 .  
## FireplaceQuTA          -1.763e+04  4.659e+03  -3.784 0.000159 ***
## GarageTypeA             1.774e+04  6.539e+03   2.713 0.006728 ** 
## GarageTypeBI            1.719e+04  7.156e+03   2.402 0.016395 *  
## GarageTypeBM            1.422e+04  8.279e+03   1.718 0.086021 .  
## GarageTypeCP            5.891e+03  9.526e+03   0.618 0.536416    
## GarageTypeD             1.255e+04  6.554e+03   1.915 0.055707 .  
## GarageTypeNone         -2.353e+05  9.418e+04  -2.499 0.012557 *  
## GarageYrBlt            -1.305e+02  4.873e+01  -2.679 0.007449 ** 
## GarageFinishNone               NA         NA      NA       NA    
## GarageFinishRFn        -7.442e+03  1.691e+03  -4.401 1.14e-05 ***
## GarageFinishUnf        -5.400e+03  2.007e+03  -2.691 0.007192 ** 
## GarageCars              9.267e+03  1.911e+03   4.850 1.34e-06 ***
## GarageArea              1.634e+01  6.968e+00   2.344 0.019169 *  
## GarageQualTA            3.165e+03  2.824e+03   1.121 0.262591    
## PavedDriveP            -1.757e+03  4.410e+03  -0.398 0.690339    
## PavedDriveY             1.357e+03  2.834e+03   0.479 0.632227    
## WoodDeckSF              1.186e+01  5.013e+00   2.366 0.018095 *  
## OpenPorchSF            -8.345e+00  9.283e+00  -0.899 0.368790    
## EnclosedPorch           1.303e+01  1.028e+01   1.268 0.205029    
## X3SsnPorch              3.304e+01  2.217e+01   1.490 0.136281    
## ScreenPorch             3.349e+01  1.008e+01   3.322 0.000910 ***
## PoolArea                1.516e+03  1.493e+02  10.153  < 2e-16 ***
## PoolQCNo Pool           9.924e+05  9.212e+04  10.773  < 2e-16 ***
## PoolQCTA                1.287e+05  3.064e+04   4.200 2.80e-05 ***
## FenceGdWo              -1.410e+03  4.091e+03  -0.345 0.730437    
## FenceMnPrv             -3.792e+03  3.213e+03  -1.180 0.238039    
## FenceMnWw              -3.097e+03  9.594e+03  -0.323 0.746854    
## FenceNo Fence          -3.780e+03  2.890e+03  -1.308 0.190940    
## MiscFeatureNo Feature   1.257e+03  3.037e+03   0.414 0.679002    
## MoSold                 -4.768e+01  2.062e+02  -0.231 0.817148    
## YrSold                 -4.858e+02  4.362e+02  -1.114 0.265620    
## SaleTypeOther          -7.106e+03  3.553e+03  -2.000 0.045670 *  
## SaleTypeWarranty Deed  -6.050e+03  2.470e+03  -2.449 0.014414 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 23040 on 1791 degrees of freedom
## Multiple R-squared:  0.901,  Adjusted R-squared:  0.8911 
## F-statistic: 90.57 on 180 and 1791 DF,  p-value: < 2.2e-16
```

```
mod_linear_olsrr <- ols_step_backward_p(mod_linear_intial, p_val = 0.05, progress = TRU
E)
```

```
## Backward Elimination Method
## ---------------------------
##
## Candidate Terms:
##
## 1. MSZoning
## 2. LotFrontage
## 3. LotArea
## 4. Street
## 5. Alley
## 6. LotShape
## 7. LandContour
## 8. LotConfig
## 9. LandSlope
## 10. Neighborhood
## 11. Condition1
## 12. Condition2
## 13. BldgType
## 14. HouseStyle
## 15. YearBuilt
## 16. YearRemodAdd
## 17. RoofStyle
## 18. RoofMatl
## 19. Exterior1st
## 20. Exterior2nd
## 21. MasVnrType
## 22. MasVnrArea
## 23. ExterQual
## 24. ExterCond
## 25. Foundation
## 26. BsmtQual
## 27. BsmtCond
## 28. BsmtExposure
## 29. BsmtFinType1
## 30. BsmtFinSF1
## 31. BsmtFinType2
## 32. BsmtFinSF2
## 33. BsmtUnfSF
## 34. TotalBsmtSF
## 35. Heating
## 36. HeatingQC
## 37. CentralAir
## 38. Electrical
## 39. X1stFlrSF
## 40. X2ndFlrSF
## 41. LowQualFinSF
## 42. GrLivArea
## 43. BsmtFullBath
## 44. BsmtHalfBath
## 45. FullBath
## 46. HalfBath
## 47. BedroomAbvGr
## 48. KitchenAbvGr
## 49. KitchenQual
## 50. TotRmsAbvGrd
## 51. Functional
## 52. Fireplaces
## 53. FireplaceQu
## 54. GarageType
## 55. GarageYrBlt
## 56. GarageFinish
## 57. GarageCars
## 58. GarageArea
## 59. GarageQual
## 60. PavedDrive
## 61. WoodDeckSF
## 62. OpenPorchSF
## 63. EnclosedPorch
## 64. X3SsnPorch
## 65. ScreenPorch
## 66. PoolArea
## 67. PoolQC
## 68. Fence
## 69. MiscFeature
## 70. MoSold
## 71. YrSold
## 72. SaleType
##
##
## Variables Removed:
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => Street
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => MoSold
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => RoofMatl
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => TotRmsAbvGrd
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => BedroomAbvGr
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => Electrical
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => Heating
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => MiscFeature
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => HalfBath
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => CentralAir
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => LotShape
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => HouseStyle
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => BsmtCond
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => BsmtHalfBath
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => MasVnrType
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => MasVnrArea
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => Fence
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => PavedDrive
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => MSZoning
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => BsmtFinType2
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => Foundation
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => Alley
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => LotArea
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => OpenPorchSF
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => LotConfig
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => YrSold
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => RoofStyle
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => Condition2
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => EnclosedPorch
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => X3SsnPorch
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => GarageQual
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
## Warning in b * sx: longer object length is not a multiple of shorter object
## length
```

```
## => ExterCond
```

```
## Note: model has aliased coefficients
##       sums of squares computed by model comparison
```

```
##
## No more variables to be removed.
```

```
summary(mod_linear_olsrr)
```

```
##         Length Class      Mode
## metrics  9     data.frame list
## model   13     lm         list
## others   1     -none-     list
```

```
mod_linear_final <- mod_linear_olsrr$model
summary(mod_linear_final)
```

```
##
## Call:
## lm(formula = paste(response, "~", paste(c(include, cterms), collapse = " + ")),
##     data = l)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -311695  -10836     -12   10820  130403
##
## Coefficients: (5 not defined because of singularities)
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -1.380e+06  1.613e+05  -8.556  < 2e-16 ***
## LotFrontage             -1.126e+02  3.877e+01  -2.904 0.003722 **
## LandContourHLS           1.915e+04  4.065e+03   4.712 2.63e-06 ***
## LandContourLow           6.996e+03  5.042e+03   1.388 0.165435
## LandContourLvl           1.233e+04  2.949e+03   4.181 3.03e-05 ***
## LandSlopeNot Gtl         1.027e+04  3.157e+03   3.252 0.001168 **
## NeighborhoodBlueste     -3.016e+03  1.223e+04  -0.247 0.805276
## NeighborhoodBrDale       1.363e+04  8.777e+03   1.553 0.120491
## NeighborhoodBrkSide     -7.414e+03  7.324e+03  -1.012 0.311506
## NeighborhoodClearCr      1.071e+04  7.804e+03   1.372 0.170333
## NeighborhoodCollgCr      3.121e+03  6.344e+03   0.492 0.622784
## NeighborhoodCrawfor      2.171e+04  7.096e+03   3.059 0.002254 **
## NeighborhoodEdwards     -1.824e+04  6.751e+03  -2.702 0.006965 **
## NeighborhoodGilbert     -6.415e+03  6.523e+03  -0.983 0.325541
## NeighborhoodGreens       1.243e+04  1.343e+04   0.926 0.354797
## NeighborhoodIDOTRR      -1.205e+04  7.549e+03  -1.597 0.110533
## NeighborhoodMeadowV     -1.432e+04  7.940e+03  -1.804 0.071419 .
## NeighborhoodMitchel     -6.924e+03  6.783e+03  -1.021 0.307530
## NeighborhoodNAmes       -6.920e+03  6.657e+03  -1.040 0.298703
## NeighborhoodNoRidge      4.115e+04  7.134e+03   5.768 9.40e-09 ***
## NeighborhoodNPkVill      1.416e+04  8.923e+03   1.587 0.112595
## NeighborhoodNridgHt      4.279e+04  6.551e+03   6.532 8.36e-11 ***
## NeighborhoodNWAmes      -2.902e+03  6.855e+03  -0.423 0.672064
## NeighborhoodOldTown     -1.385e+04  7.103e+03  -1.950 0.051347 .
## NeighborhoodSawyer      -6.016e+03  6.874e+03  -0.875 0.381581
## NeighborhoodSawyerW     -1.484e+03  6.612e+03  -0.224 0.822474
## NeighborhoodSomerst      2.036e+04  6.296e+03   3.234 0.001241 **
## NeighborhoodStoneBr      4.217e+04  7.359e+03   5.730 1.17e-08 ***
## NeighborhoodSWISU       -1.208e+04  7.936e+03  -1.522 0.128178
## NeighborhoodTimber       1.179e+04  6.912e+03   1.705 0.088300 .
## NeighborhoodVeenker      3.512e+04  9.046e+03   3.882 0.000107 ***
## Condition1Feedr          2.651e+03  4.016e+03   0.660 0.509246
## Condition1Norm           1.154e+04  3.341e+03   3.453 0.000567 ***
## Condition1PosA           2.593e+04  7.001e+03   3.703 0.000219 ***
## Condition1PosN           1.298e+04  5.640e+03   2.301 0.021512 *
## Condition1RRAe           2.077e+02  6.489e+03   0.032 0.974463
## Condition1RRAn           3.297e+03  5.473e+03   0.602 0.546950
## Condition1RRNe           1.628e+03  1.106e+04   0.147 0.882978
## Condition1RRNn           8.799e+03  1.110e+04   0.793 0.427956
## BldgType2fmCon          -5.139e+03  4.473e+03  -1.149 0.250782
## BldgTypeDuplex          -9.638e+03  4.856e+03  -1.985 0.047326 *
## BldgTypeTwnhs          -3.403e+04  4.435e+03  -7.674 2.68e-14 ***
## BldgTypeTwnhsE         -2.741e+04  2.872e+03  -9.545  < 2e-16 ***
## YearBuilt               1.588e+02  5.222e+01   3.041 0.002394 **
## YearRemodAdd            2.268e+02  4.045e+01   5.607 2.36e-08 ***
## Exterior1stMetalSd     -3.585e+03  6.875e+03  -0.522 0.602066
## Exterior1stOther        6.388e+03  3.482e+03   1.835 0.066720 .
## Exterior1stVinylSd     -1.260e+04  7.060e+03  -1.785 0.074384 .
## Exterior1stWd Sdng     -1.980e+03  4.389e+03  -0.451 0.652033
## Exterior2ndMetalSd      8.563e+03  6.872e+03   1.246 0.212913
## Exterior2ndOther       -1.972e+03  3.451e+03  -0.571 0.567786
## Exterior2ndVinylSd      1.548e+04  7.107e+03   2.178 0.029547 *
## Exterior2ndWd Sdng      6.260e+03  4.459e+03   1.404 0.160522
## ExterQualFa            -2.368e+04  7.579e+03  -3.125 0.001806 **
## ExterQualGd            -1.119e+04  3.972e+03  -2.818 0.004881 **
## ExterQualTA            -2.048e+04  4.378e+03  -4.677 3.13e-06 ***
## BsmtQualFa             -1.946e+04  4.734e+03  -4.111 4.10e-05 ***
## BsmtQualGd             -1.771e+04  2.727e+03  -6.492 1.08e-10 ***
## BsmtQualNo Basement    -2.751e+04  6.527e+03  -4.215 2.62e-05 ***
## BsmtQualTA             -1.531e+04  3.338e+03  -4.587 4.81e-06 ***
## BsmtExposureGd          1.083e+04  2.479e+03   4.367 1.33e-05 ***
## BsmtExposureMn         -5.892e+03  2.411e+03  -2.444 0.014610 *
## BsmtExposureNo Basement        NA         NA      NA       NA
## BsmtExposureNo Exposure -4.508e+03  1.709e+03  -2.637 0.008426 **
## BsmtFinType1BLQ        -2.425e+02  2.247e+03  -0.108 0.914074
## BsmtFinType1GLQ         4.403e+03  2.027e+03   2.172 0.030014 *
## BsmtFinType1LwQ        -5.687e+03  2.783e+03  -2.043 0.041146 *
## BsmtFinType1No Basement        NA         NA      NA       NA
## BsmtFinType1Rec        -2.398e+03  2.293e+03  -1.046 0.295675
## BsmtFinType1Unf        -3.488e+03  2.313e+03  -1.508 0.131748
## BsmtFinSF1              1.085e+01  3.660e+00   2.964 0.003075 **
## BsmtFinSF2              1.052e+01  4.839e+00   2.173 0.029900 *
## BsmtUnfSF               8.669e+00  3.394e+00   2.554 0.010729 *
## TotalBsmtSF                    NA         NA      NA       NA
## HeatingQCFa            -9.912e+03  3.226e+03  -3.072 0.002155 **
## HeatingQCGd            -1.881e+03  1.657e+03  -1.135 0.256552
## HeatingQCTA            -4.831e+03  1.610e+03  -3.000 0.002740 **
## X1stFlrSF               4.509e+01  3.556e+00  12.682  < 2e-16 ***
## X2ndFlrSF               4.055e+01  2.048e+00  19.801  < 2e-16 ***
## LowQualFinSF            1.584e+01  1.186e+01   1.336 0.181847
## GrLivArea                      NA         NA      NA       NA
## BsmtFullBath            7.627e+03  1.435e+03   5.316 1.19e-07 ***
## FullBath                4.151e+03  1.575e+03   2.635 0.008474 **
## KitchenAbvGr           -1.253e+04  4.232e+03  -2.961 0.003110 **
## KitchenQualFa          -2.714e+04  5.059e+03  -5.364 9.14e-08 ***
## KitchenQualGd          -2.289e+04  3.132e+03  -7.307 4.04e-13 ***
## KitchenQualTA          -2.700e+04  3.457e+03  -7.810 9.46e-15 ***
## FunctionalMaj2         -2.492e+04  1.202e+04  -2.073 0.038282 *
## FunctionalMin1          3.770e+03  7.911e+03   0.476 0.633784
## FunctionalMin2         -1.472e+03  7.770e+03  -0.189 0.849799
## FunctionalMod          -3.358e+03  8.544e+03  -0.393 0.694328
## FunctionalTyp           1.288e+04  7.080e+03   1.819 0.069083 .
## Fireplaces              8.399e+03  2.161e+03   3.886 0.000106 ***
## FireplaceQuFa           -2.154e+04  5.565e+03  -3.871 0.000112 ***
## FireplaceQuGd           -1.556e+04  4.452e+03  -3.495 0.000486 ***
## FireplaceQuNo Fireplace -1.530e+04  5.156e+03  -2.967 0.003041 **
## FireplaceQuPo           -1.282e+04  6.540e+03  -1.960 0.050145 .
## FireplaceQuTA           -1.714e+04  4.567e+03  -3.753 0.000180 ***
## GarageTypeA              1.699e+04  6.257e+03   2.716 0.006667 **
## GarageTypeBI             1.574e+04  6.750e+03   2.332 0.019794 *
## GarageTypeBM             1.193e+04  7.967e+03   1.498 0.134372
## GarageTypeCP             2.455e+03  9.143e+03   0.269 0.788321
## GarageTypeD              1.127e+04  6.275e+03   1.796 0.072728 .
## GarageTypeNone          -2.317e+05  8.915e+04  -2.599 0.009426 **
## GarageYrBlt             -1.269e+02  4.595e+01  -2.761 0.005819 **
## GarageFinishNone                NA         NA      NA       NA
## GarageFinishRFn         -7.585e+03  1.642e+03  -4.618 4.13e-06 ***
## GarageFinishUnf         -5.435e+03  1.940e+03  -2.802 0.005129 **
## GarageCars               9.235e+03  1.822e+03   5.069 4.40e-07 ***
## GarageArea               1.753e+01  6.689e+00   2.621 0.008845 **
## WoodDeckSF               1.126e+01  4.796e+00   2.347 0.019011 *
## ScreenPorch              3.098e+01  9.800e+00   3.161 0.001600 **
## PoolArea                 1.504e+03  1.393e+02  10.802  < 2e-16 ***
## PoolQCNo Pool            9.862e+05  8.637e+04  11.419  < 2e-16 ***
## PoolQCTA                 1.292e+05  2.935e+04   4.401 1.14e-05 ***
## SaleTypeOther           -7.814e+03  3.425e+03  -2.282 0.022625 *
## SaleTypeWarranty Deed   -5.956e+03  2.365e+03  -2.518 0.011883 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22930 on 1860 degrees of freedom
## Multiple R-squared:  0.8983, Adjusted R-squared:  0.8922
## F-statistic: 147.9 on 111 and 1860 DF,  p-value: < 2.2e-16
```

```
# out of sample Rsqd calc
out_of_sample_predictions <- predict(mod_linear_final, newdata = test_ames)

# calculate out-of-sample R-squared
out_of_sample_r_squared <- 1 - (sum((test_ames$SalePrice - out_of_sample_predictions)^2)
/
                                sum((test_ames$SalePrice - mean(test_ames$SalePrice))^
2))
out_of_sample_r_squared
```

```
## [1] 0.825442
```

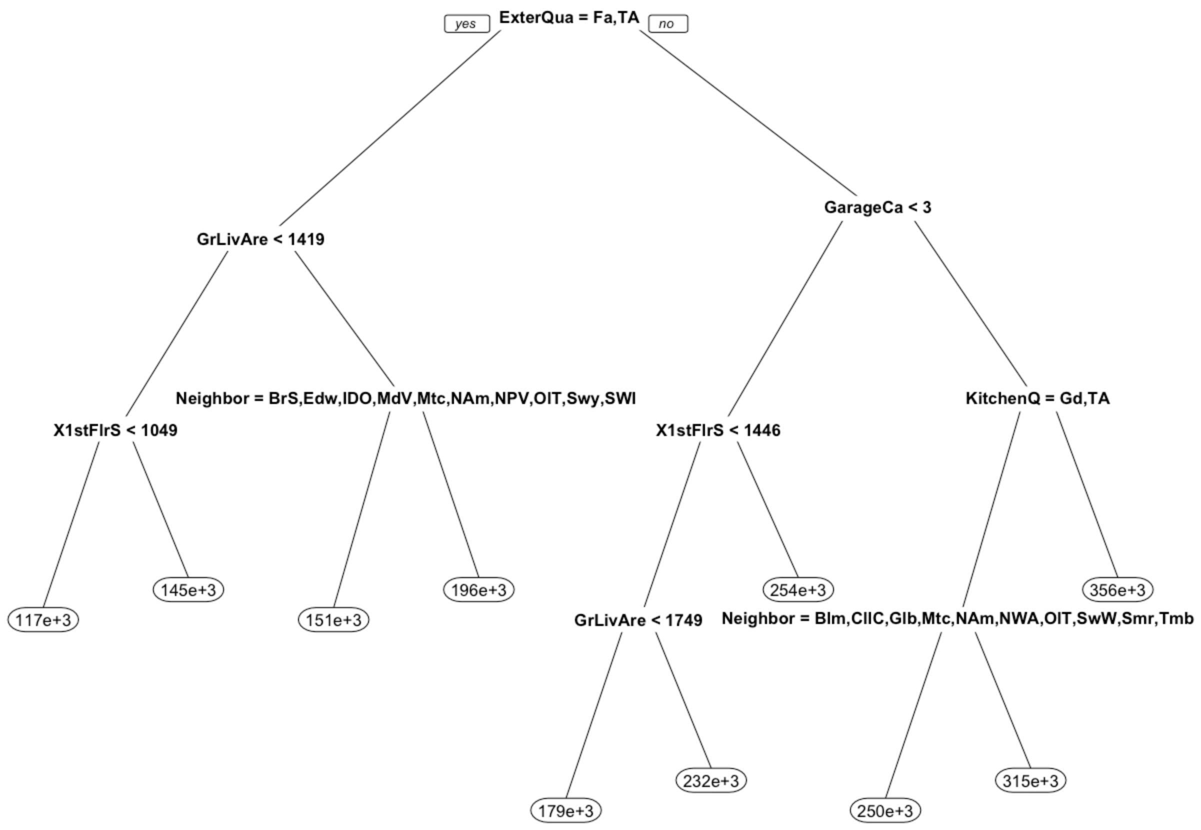
For mod_linear_final: In-sample R-squared (taken from outputted summary): 0.8983 Out-of-sample R-squared: 0.8254482

Part b

```
set.seed(15072)
library(rpart)
modelcart = rpart(data = train_ames, SalePrice ~ .)
modelcart
```

```
## n= 1972
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 1972 9.608319e+12 178139.2
##    2) ExterQual=Fa,TA 1245 1.900199e+12 143154.8
##      4) GrLivArea< 1419 785 5.444631e+11 127737.4
##        8) X1stFlrSF< 1049 481 2.328889e+11 116714.7 *
##        9) X1stFlrSF>=1049 304 1.606640e+11 145178.0 *
##      5) GrLivArea>=1419 460 8.507225e+11 169464.9
##       10) Neighborhood=BrkSide,Edwards,IDOTRR,MeadowV,Mitchel,NAmes,NPkVill,OldTown,S
awyer,SWISU 274 3.673466e+11 151150.2 *
##       11) Neighborhood=ClearCr,CollgCr,Crawfor,Gilbert,NridgHt,NWAmes,SawyerW,Somers
t,Timber,Veenker 186 2.560787e+11 196444.6 *
##    3) ExterQual=Ex,Gd 727 3.574869e+12 238050.7
##      6) GarageCars< 2.5 510 1.269908e+12 209839.7
##       12) X1stFlrSF< 1446 373 5.675778e+11 193570.8
##         24) GrLivArea< 1748.5 269 2.177148e+11 178686.4 *
##         25) GrLivArea>=1748.5 104 1.361211e+11 232069.8 *
##       13) X1stFlrSF>=1446 137 3.348141e+11 254134.0 *
##      7) GarageCars>=2.5 217 9.451414e+11 304353.0
##       14) KitchenQual=Gd,TA 142 3.937172e+11 277137.0
##         28) Neighborhood=Blmngtn,CollgCr,Gilbert,Mitchel,NAmes,NWAmes,OldTown,Sawyer
W,Somerst,Timber 83 1.250205e+11 249986.9 *
##         29) Neighborhood=NoRidge,NridgHt,StoneBr,Veenker 59 1.214454e+11 315331.3 *
##       15) KitchenQual=Ex 75 2.471010e+11 355881.9 *
```

```
library(rpart.plot)
prp(modelcart)
```



```
# in-sample predictions
in_sample_predictions <- predict(modelcart, newdata = train_ames)

# calculate in-sample R-squared
in_sample_r_squared <- 1 - (sum((train_ames$SalePrice - in_sample_predictions)^2) /
                            sum((train_ames$SalePrice - mean(train_ames$SalePrice))^2))

# out-of-sample predictions
out_of_sample_predictions <- predict(modelcart, newdata = test_ames)

# calculate out-of-sample R-squared
out_of_sample_r_squared <- 1 - (sum((test_ames$SalePrice - out_of_sample_predictions)^2)
/
                                sum((test_ames$SalePrice - mean(test_ames$SalePrice))^
2))

cat("In-sample R²:", round(in_sample_r_squared, 4), "\n")
```

```
## In-sample R²: 0.7711
```

```
cat("Out-of-sample R²:", round(out_of_sample_r_squared, 4), "\n")
```

```
## Out-of-sample R²: 0.691
```

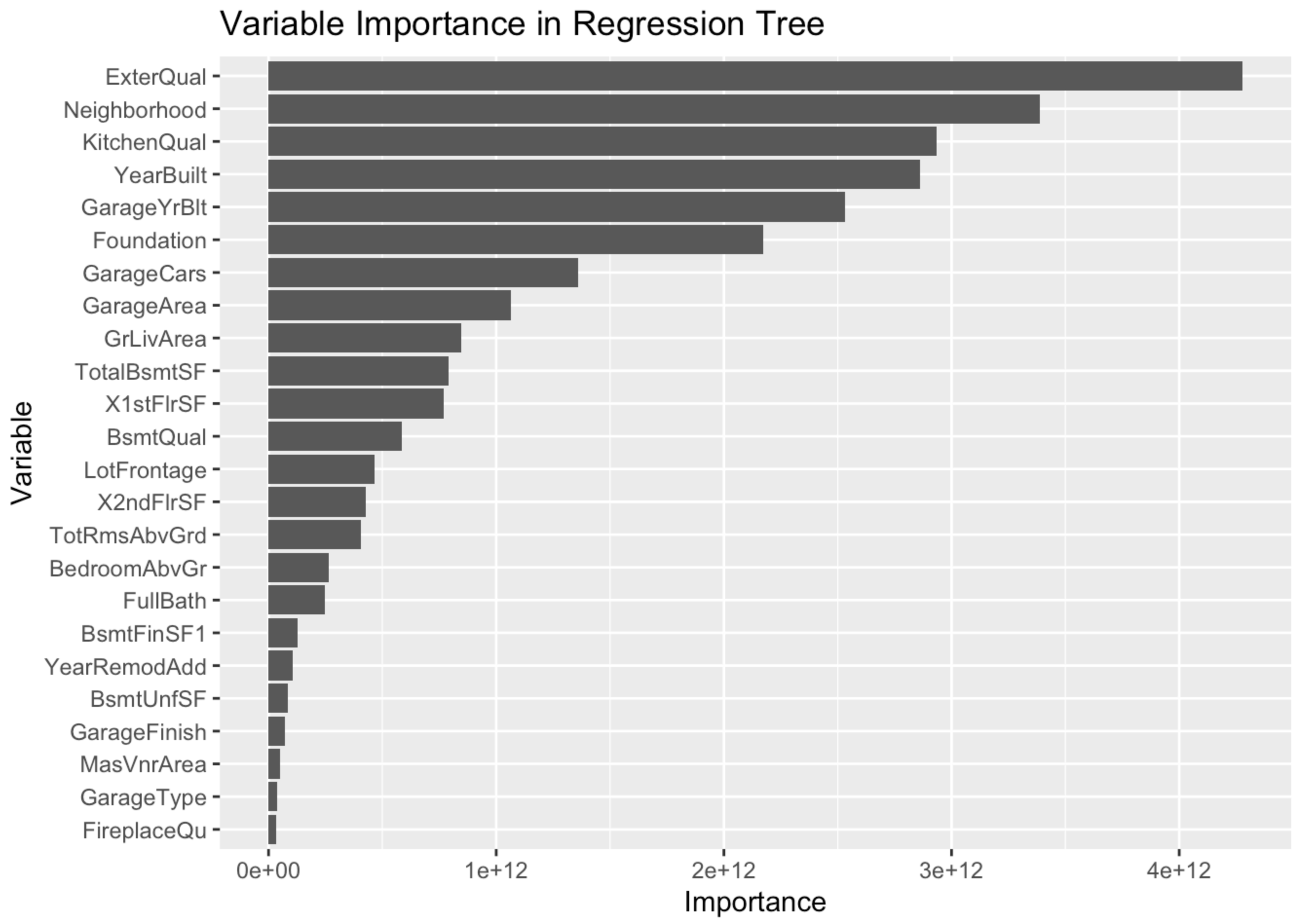
```
set.seed(15072)
importance_scores <- modelcart$variable.importance
print(importance_scores)
```

```
##    ExterQual Neighborhood  KitchenQual    YearBuilt  GarageYrBlt   Foundation
## 4.279327e+12 3.389753e+12 2.936123e+12 2.862491e+12 2.532263e+12 2.174458e+12
##   GarageCars   GarageArea    GrLivArea  TotalBsmtSF    X1stFlrSF     BsmtQual
## 1.359819e+12 1.065296e+12 8.466353e+11 7.909058e+11 7.674546e+11 5.851225e+11
##  LotFrontage    X2ndFlrSF TotRmsAbvGrd BedroomAbvGr     FullBath   BsmtFinSF1
## 4.642459e+11 4.272400e+11 4.075687e+11 2.623970e+11 2.459194e+11 1.287648e+11
## YearRemodAdd    BsmtUnfSF GarageFinish   MasVnrArea   GarageType  FireplaceQu
## 1.048548e+11 8.584321e+10 7.209965e+10 4.991570e+10 3.772757e+10 3.493857e+10
```

```
sorted_importance <- sort(importance_scores, decreasing = TRUE)
print(sorted_importance)
```

```
##    ExterQual Neighborhood  KitchenQual    YearBuilt  GarageYrBlt   Foundation
## 4.279327e+12 3.389753e+12 2.936123e+12 2.862491e+12 2.532263e+12 2.174458e+12
##   GarageCars   GarageArea    GrLivArea  TotalBsmtSF    X1stFlrSF     BsmtQual
## 1.359819e+12 1.065296e+12 8.466353e+11 7.909058e+11 7.674546e+11 5.851225e+11
##  LotFrontage    X2ndFlrSF TotRmsAbvGrd BedroomAbvGr     FullBath   BsmtFinSF1
## 4.642459e+11 4.272400e+11 4.075687e+11 2.623970e+11 2.459194e+11 1.287648e+11
## YearRemodAdd    BsmtUnfSF GarageFinish   MasVnrArea   GarageType  FireplaceQu
## 1.048548e+11 8.584321e+10 7.209965e+10 4.991570e+10 3.772757e+10 3.493857e+10
```

```
library(ggplot2)
importance_df <- as.data.frame(sorted_importance)
importance_df$variable <- rownames(importance_df)
ggplot(importance_df, aes(x = reorder(variable, sorted_importance), y = sorted_importanc
e)) +
  geom_col() +
  coord_flip() +
  labs(x = "Variable", y = "Importance", title = "Variable Importance in Regression Tre
e")
```



In sample R^2 is 0.7711155 and out of sample R^2 is 0.6909755. We have included our decision tree visualization above, and we can see from this plot of feature importances that the 5 most important variables are ExterQual, Neighborhood, KitchenQual, YearBuilt, and GarageYrBlt. This makes sense because the initial splits in our tree do occur based on ExterQua, GrLivAre, and GarageCa, and Neighbor shows up as well in following splits. These features must provide the most information gain in the tree, hence they are some of the most important features that tell us more about how to predict SalePrice based on other attributes provided.

Part c

```
set.seed(15072)
coef(mod_linear_final
     )["CentralAir"]
```

```
## <NA>
##   NA
```

In order to see if it is worth it to have central air installed in order to increase the value of her home, we would want to compare the cost of installation ($15,000) to the predicted value added when a home has central air but all other factors remain constant. If the predicted value add is more than the cost, then yes it is worth it. Else, it is not going to be enough of a reason to spend the cost on the install. When we try to find the coefficient for "CentralAir" from our initial linear model, we get NA. When we try to see if "CentralAir" was one of the important features in our plot of Important Features in our decision tree, we see it doesn't show up. Meaning, "CentralAir" is not being valued in either of our models - the noninclusion suggests 0 value add. One thing I noticed was that, during the construction of models in part a, I was getting the error of aliased coefficients. I investigated further and found that a model has “aliased coefficients” when there is linear dependency among its predictor variables, meaning one or more predictors can be perfectly expressed as a linear combination of others, resulting in redundant information and making it impossible to uniquely estimate their coefficients. It indicates potential multicollinearity issues that require addressing by removing redundant variables or using other modeling techniques. However, we were told to assume no multicollinearity, which is obviously not the case in this dataset.

Part d

```
set.seed(15072)

# defining the cp values to evaluate
cp_values <- c(5e-6, 5e-5, 5e-4, 5e-3, 5e-2)
tune_grid <- expand.grid(cp = cp_values)

# set up 10-fold cross-validation
ctrl <- trainControl(method = "cv", number = 10)

# perform cross-validation to find the best cp
set.seed(15072)
tree_model_cv <- train(
  SalePrice ~ .,
  data = train_ames,
  method = "rpart",
  trControl = ctrl,
  tuneGrid = tune_grid
)

# result of cross validation
print(tree_model_cv)
```

```
## CART
##
## 1972 samples
##   72 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1773, 1776, 1775, 1774, 1775, 1776, ...
## Resampling results across tuning parameters:
##
##   cp     RMSE      Rsquared   MAE
##   5e-06  32402.72  0.7888318  22046.39
##   5e-05  32381.31  0.7891214  22008.86
##   5e-04  32280.79  0.7894255  21944.28
##   5e-03  35459.33  0.7410871  25650.75
##   5e-02  44703.89  0.5875329  33500.24
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 5e-04.
```

```
set.seed(15072)

# reporting the best cp value selected by cross-validation based on lowest rmse
best_cp <- tree_model_cv$bestTune$cp
cat("The optimal cp value is:", best_cp, "\n")
```

```
## The optimal cp value is: 5e-04
```

```
# define the final model using the best cp
final_model <- rpart(SalePrice ~ ., data = train_ames, cp = best_cp)
print(final_model)
```

```
## n= 1972
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##   1) root 1972 9.608319e+12 178139.20
##     2) ExterQual=Fa,TA 1245 1.900199e+12 143154.80
##       4) GrLivArea< 1419 785 5.444631e+11 127737.40
##         8) X1stFlrSF< 1049 481 2.328889e+11 116714.70
##          16) Neighborhood=BrDale,BrkSide,Edwards,IDOTRR,MeadowV,OldTown,SawyerW,SWISU
265 1.076367e+11 106900.90
##            32) GrLivArea< 1146 170 5.917118e+10  99635.68
##              64) TotalBsmtSF< 685 87 2.257950e+10  92467.70
##               128) GrLivArea< 952 40 5.243056e+09  84340.00 *
##               129) GrLivArea>=952 47 1.244523e+10  99384.89 *
##              65) TotalBsmtSF>=685 83 2.743616e+10 107149.10
##               130) PavedDrive=N,P 24 7.332318e+09  93667.71 *
##               131) PavedDrive=Y 59 1.396754e+10 112633.10 *
##            33) GrLivArea>=1146 95 2.343486e+10 119901.90 *
##          17) Neighborhood=Blueste,ClearCr,CollgCr,Crawfor,Gilbert,Mitchel,NAmes,NPkVi
ll,NWAmes,Sawyer,Somerst,Timber 216 6.841824e+10 128754.70
##            34) HouseStyle=1.5Unf,1Story 153 3.940272e+10 123441.70
##              68) TotalBsmtSF< 776.5 19 2.222815e+09 101400.00 *
##              69) TotalBsmtSF>=776.5 134 2.664019e+10 126567.00 *
##            35) HouseStyle=1.5Fin,2Story,SFoyer,SLvl 63 1.420769e+10 141657.80 *
##         9) X1stFlrSF>=1049 304 1.606640e+11 145178.00
##          18) Neighborhood=BrkSide,Crawfor,Edwards,IDOTRR,MeadowV,NAmes,NPkVill,OldTow
n,Sawyer,SawyerW,SWISU 236 9.176543e+10 139985.40
##            36) YearBuilt< 1953.5 41 1.257992e+10 121856.60
##              72) LotFrontage< 79.5 31 4.707429e+09 114568.40 *
##              73) LotFrontage>=79.5 10 1.121225e+09 144450.00 *
##            37) YearBuilt>=1953.5 195 6.287762e+10 143797.10
##              74) TotalBsmtSF< 472 11 3.389282e+09 112027.30 *
##              75) TotalBsmtSF>=472 184 4.772208e+10 145696.40
##               150) X1stFlrSF< 1187.5 106 2.130203e+10 140895.90 *
##               151) X1stFlrSF>=1187.5 78 2.065788e+10 152220.00 *
##          19) Neighborhood=ClearCr,CollgCr,Gilbert,Mitchel,NWAmes,Timber 68 4.045119e+
10 163199.20
##            38) BsmtExposure=Av,Mn,No Exposure 61 2.654873e+10 159246.70
##              76) GarageArea< 334.5 8 3.448840e+09 134800.00 *
##              77) GarageArea>=334.5 53 1.759708e+10 162936.80 *
##            39) BsmtExposure=Gd 7 4.644937e+09 197642.90 *
##       5) GrLivArea>=1419 460 8.507225e+11 169464.90
##        10) Neighborhood=BrkSide,Edwards,IDOTRR,MeadowV,Mitchel,NAmes,NPkVill,OldTown,
Sawyer,SWISU 274 3.673466e+11 151150.20
##          20) LotArea< 12561.5 221 1.569281e+11 143602.90
##            40) GarageQual=Other 50 2.758929e+10 120574.40 *
##            41) GarageQual=TA 171 9.507018e+10 150336.30
##              82) YearRemodAdd< 1961 59 2.011788e+10 138366.90 *
##              83) YearRemodAdd>=1961 112 6.204682e+10 156641.60
##               166) GrLivArea< 1932.5 88 3.678549e+10 151508.70
##                 332) BsmtFinSF1< 467 54 1.588062e+10 143912.30
##                    664) LotFrontage< 84 47 9.861887e+09 140184.30 *
##                    665) LotFrontage>=84 7 9.798371e+08 168942.90 *
##                  333) BsmtFinSF1>=467 34 1.283974e+10 163573.50 *
##                167) GrLivArea>=1932.5 24 1.444138e+10 175462.50
##                  334) GarageYrBlt>=1957 17 6.755721e+09 165641.20 *
##                  335) GarageYrBlt< 1957 7 2.063509e+09 199314.30 *
##          21) LotArea>=12561.5 53 1.453373e+11 182621.20
##            42) KitchenQual=Fa,TA 41 8.432780e+10 167486.00
##              84) Fireplaces< 1.5 30 3.194658e+10 148714.20
##               168) Fireplaces< 0.5 9 7.021340e+09 119766.70 *
##               169) Fireplaces>=0.5 21 1.415150e+10 161120.20
##                 338) GrLivArea< 1657 9 6.210722e+09 143444.40 *
##                 339) GrLivArea>=1657 12 3.019944e+09 174377.10 *
##              85) Fireplaces>=1.5 11 1.297864e+10 218681.80 *
##            43) KitchenQual=Ex,Gd 12 1.952767e+10 234333.30 *
##        11) Neighborhood=ClearCr,CollgCr,Crawfor,Gilbert,NridgHt,NWAmes,SawyerW,Somers
t,Timber,Veenker 186 2.560787e+11 196444.60
##          22) GrLivArea< 2093.5 155 1.341352e+11 187091.70
##            44) BsmtFinSF1< 593.5 99 5.366151e+10 176118.30
##              88) KitchenAbvGr>=1.5 8 1.897482e+09 131600.90 *
##              89) KitchenAbvGr< 1.5 91 3.451583e+10 180031.90
##               178) TotalBsmtSF< 786.5 33 1.196113e+10 170088.00
##                 356) BsmtFinSF1< 217.5 20 4.757086e+09 159856.00 *
##                 357) BsmtFinSF1>=217.5 13 1.888763e+09 185829.60 *
##               179) TotalBsmtSF>=786.5 58 1.743503e+10 185689.70 *
##            45) BsmtFinSF1>=593.5 56 4.747746e+10 206491.20
##              90) Neighborhood=Gilbert,NWAmes,SawyerW,Veenker 29 8.277487e+09 190848.8
0 *
##              91) Neighborhood=ClearCr,CollgCr,Crawfor,Somerst,Timber 27 2.448267e+10
223292.30
##               182) Exterior1st=HdBoard,VinylSd,Wd Sdng 18 6.053178e+09 208951.60 *
##               183) Exterior1st=Other 9 7.324022e+09 251973.80 *
##          23) GrLivArea>=2093.5 31 4.059107e+10 243208.80
##            46) Neighborhood=ClearCr,Crawfor,Gilbert,Timber 19 1.508205e+10 226305.30
*
##            47) Neighborhood=CollgCr,NridgHt,NWAmes,SawyerW 12 1.148445e+10 269972.80
*
##     3) ExterQual=Ex,Gd 727 3.574869e+12 238050.70
##       6) GarageCars< 2.5 510 1.269908e+12 209839.70
##        12) X1stFlrSF< 1446 373 5.675778e+11 193570.80
##          24) GrLivArea< 1748.5 269 2.177148e+11 178686.40
##            48) X1stFlrSF< 1274 194 1.194392e+11 169870.00
##              96) Neighborhood=BrkSide,ClearCr,Edwards,Mitchel,NAmes,OldTown,Sawyer,SW
ISU 38 1.386876e+10 140578.90 *
##              97) Neighborhood=Blmngtn,Blueste,CollgCr,Crawfor,Gilbert,Greens,NridgHt,
NWAmes,SawyerW,Somerst,StoneBr,Timber,Veenker 156 6.502606e+10 177005.00
##               194) GrLivArea< 1204 37 8.034938e+09 157738.70 *
##               195) GrLivArea>=1204 119 3.898687e+10 182995.30
##                 390) TotalBsmtSF< 769 49 6.439587e+09 172881.50 *
##                 391) TotalBsmtSF>=769 70 2.402650e+10 190075.00 *
##            49) X1stFlrSF>=1274 75 4.419064e+10 201491.50
##              98) Neighborhood=Blmngtn,CollgCr,Gilbert,NAmes,NWAmes,SawyerW 33 9.64521
```

```
3e+09 186641.80 *
##              99) Neighborhood=ClearCr,Greens,Mitchel,NridgHt,Somerst,StoneBr,Timber,V
eenker 42 2.155091e+10 213159.10
##               198) GarageFinish=RFn,Unf 32 6.788699e+09 204612.60 *
##               199) GarageFinish=Fin 10 4.945231e+09 240508.00 *
##          25) GrLivArea>=1748.5 104 1.361211e+11 232069.80
##            50) TotalBsmtSF< 1052.5 69 6.478386e+10 218543.70
##             100) Neighborhood=CollgCr,Edwards,Gilbert,NoRidge,NWAmes,OldTown,Sawyer,S
awyerW,SWISU,Timber 59 3.856651e+10 211397.60
##               200) BsmtQual=Fa,No Basement,TA 7 5.305994e+09 165628.60 *
##               201) BsmtQual=Gd 52 1.662296e+10 217558.80 *
##             101) Neighborhood=Crawfor,Somerst 10 5.427956e+09 260705.80 *
##            51) TotalBsmtSF>=1052.5 35 3.382601e+10 258735.60
##             102) GrLivArea< 2184 20 7.177199e+09 240141.00 *
##             103) GrLivArea>=2184 15 1.051333e+10 283528.50 *
##        13) X1stFlrSF>=1446 137 3.348141e+11 254134.00
##          26) Neighborhood=Blmngtn,CollgCr,Edwards,Mitchel,NAmes,NWAmes,OldTown,Sawyer
W,Somerst,StoneBr,Timber 99 1.529120e+11 239198.80
##            52) GrLivArea< 1592.5 47 3.941796e+10 221856.80
##             104) BsmtUnfSF>=1096.5 23 1.318671e+10 205102.20
##               208) Neighborhood=Blmngtn,CollgCr,Mitchel,NAmes,SawyerW,Timber 15 2.426
446e+09 193305.90 *
##               209) Neighborhood=Somerst 8 4.759368e+09 227220.10 *
##             105) BsmtUnfSF< 1096.5 24 1.358725e+10 237913.30
##               210) YearBuilt< 2004.5 17 3.519644e+09 227705.30 *
##               211) YearBuilt>=2004.5 7 3.993999e+09 262704.30 *
##            53) GrLivArea>=1592.5 52 8.658317e+10 254873.30
##             106) Neighborhood=CollgCr,Edwards,Mitchel,NAmes,NWAmes,OldTown,Timber 31
3.667409e+10 239100.40
##               212) BsmtExposure=Mn,No Exposure 19 1.346494e+10 226073.20 *
##               213) BsmtExposure=Av,Gd 12 1.487934e+10 259726.80 *
##             107) Neighborhood=SawyerW,Somerst,StoneBr 21 3.081187e+10 278157.10
##               214) YearRemodAdd< 1998 9 8.326120e+09 251600.00 *
##               215) YearRemodAdd>=1998 12 1.137756e+10 298075.00 *
##          27) Neighborhood=ClearCr,Crawfor,NoRidge,NridgHt,Veenker 38 1.022877e+11 293
044.00
##            54) TotalBsmtSF< 1567 13 1.464072e+10 259128.50 *
##            55) TotalBsmtSF>=1567 25 6.491780e+10 310680.00
##             110) GrLivArea< 1856.5 16 4.158005e+10 296965.00 *
##             111) GrLivArea>=1856.5 9 1.497764e+10 335062.30 *
##       7) GarageCars>=2.5 217 9.451414e+11 304353.00
##        14) KitchenQual=Gd,TA 142 3.937172e+11 277137.00
##          28) Neighborhood=Blmngtn,CollgCr,Gilbert,Mitchel,NAmes,NWAmes,OldTown,Sawyer
W,Somerst,Timber 83 1.250205e+11 249986.90
##            56) TotalBsmtSF< 1797.5 76 8.073447e+10 243560.60
##             112) GrLivArea< 2197.5 64 5.961513e+10 237688.50
##               224) BsmtFinSF1< 1145 56 4.376146e+10 232125.50
##                 448) OpenPorchSF< 22 12 5.776200e+09 208114.90 *
##                 449) OpenPorchSF>=22 44 2.918042e+10 238673.80 *
##               225) BsmtFinSF1>=1145 8 1.989257e+09 276629.80 *
##             113) GrLivArea>=2197.5 12 7.142676e+09 274878.70 *
##            57) TotalBsmtSF>=1797.5 7 7.072257e+09 319757.10 *
##          29) Neighborhood=NoRidge,NridgHt,StoneBr,Veenker 59 1.214454e+11 315331.30
##            58) TotalBsmtSF< 1743 43 5.455331e+10 301056.70
##             116) GrLivArea< 2390.5 23 1.552357e+10 282136.30
##               232) OpenPorchSF< 63 15 5.911989e+09 270826.70 *
##               233) OpenPorchSF>=63 8 4.095510e+09 303342.00 *
##             117) GrLivArea>=2390.5 20 2.132769e+10 322815.00
##               234) YearBuilt< 1995.5 9 5.237556e+09 301777.80 *
##               235) YearBuilt>=1995.5 11 8.848133e+09 340027.40 *
##            59) TotalBsmtSF>=1743 16 3.458255e+10 353694.40 *
##        15) KitchenQual=Ex 75 2.471010e+11 355881.90
##          30) BsmtUnfSF>=599 34 1.088776e+11 326350.40
##            60) Neighborhood=CollgCr,Edwards,NoRidge,Somerst 8 3.067148e+10 282116.60
*
##            61) Neighborhood=NridgHt,StoneBr,Timber 26 5.773678e+10 339960.80
##             122) LotArea< 12217.5 12 1.055214e+10 310238.30 *
##             123) LotArea>=12217.5 14 2.749692e+10 365437.10 *
##          31) BsmtUnfSF< 599 41 8.398221e+10 380371.50
##            62) BsmtFinSF1< 1270 18 3.566146e+10 347678.90 *
##            63) BsmtFinSF1>=1270 23 1.402603e+10 405957.00
##             126) GrLivArea< 1958 9 1.945442e+09 384827.70 *
##             127) GrLivArea>=1958 14 5.479498e+09 419540.20 *
```

```
# in-sample R-squared
in_sample_pred <- predict(final_model, newdata = train_ames)
SST_in <- sum((train_ames$SalePrice - mean(train_ames$SalePrice))^2)
SSR_in <- sum((in_sample_pred - mean(train_ames$SalePrice))^2)
R2_in_sample <- SSR_in / SST_in
cat("In-sample R-squared:", R2_in_sample, "\n")
```

```
## In-sample R-squared: 0.9105032
```

```
# out-of-sample R-squared from cross-validation results
out_of_sample_R2 <- max(tree_model_cv$results$Rsquared)
cat("Out-of-sample R-squared:", out_of_sample_R2, "\n")
```

```
## Out-of-sample R-squared: 0.7894255
```

RMSE was used to select the optimal model using the smallest value. The final value used for the model was cp = 5e-04 with n= 1972. This model has following stats: MAE of 21944.285, RMSE of 32280.79, in-sample R-squared: 0.9105032 , and out-of-sample R-squared of 0.7894255.

Part e

```
set.seed(15072)

# cross-validation control
control <- trainControl(method = "cv", number = 5)

# mtry tuning grid
mtry_values <- expand.grid(mtry = 1:73)

# train random forest with 80 trees (passed via ...)
rf_model_tuned <- train(
  SalePrice ~ .,
  data = train_ames,
  method = "rf",
  tuneGrid = mtry_values,
  trControl = control,
  importance = TRUE,
  ntree = 80
)

# display selected mtry
selected_mtry <- rf_model_tuned$bestTune$mtry
cat("Selected mtry value:", selected_mtry, "\n")
```

```
## Selected mtry value: 44
```

```
# in-sample R²
pred_train <- predict(rf_model_tuned, newdata = train_ames)
SSE_train <- sum((train_ames$SalePrice - pred_train)^2)
SST_train <- sum((train_ames$SalePrice - mean(train_ames$SalePrice))^2)
R2_train <- 1 - SSE_train / SST_train

# out-of-sample R² using test set mean
pred_test <- predict(rf_model_tuned, newdata = test_ames)
SSE_test <- sum((test_ames$SalePrice - pred_test)^2)
SST_test <- sum((test_ames$SalePrice - mean(test_ames$SalePrice))^2)
R2_test <- 1 - SSE_test / SST_test

cat("In-sample R²:", round(R2_train, 4), "\n")
```

```
## In-sample R²: 0.9811
```

```
cat("Out-of-sample R²:", round(R2_test, 4), "\n")
```

```
## Out-of-sample R²: 0.8733
```

Part f

Out of the four models constructed, I would recommend my model from part e - a random forest model with 80 trees, a nodesize of 25, and selected mtry value of 44. I believe this is the most robust model compared to the other linear regression and CART models. Out of all 4 models, this model had the highest out of sample R^2 at 0.8733, meaning that 87.33% of the variation in SalePrice from the test dataset could be explained by the model built on 44 variables chosen from the training dataset. The other models we looked at all had lower OOS R^2 at 0.83, 0.69, and 0.79 respectively for a, b, and d. Despite most of those models showing promising in sample R^2 (all above 0.75 and two around 0.90), they didn't perform as well with the test data, meaning that there could be some sort of overfitting occurring that isn't allowing for generalization towards unseen data. Our model in e had an in-sample R^2 of 0.9811. Also, I prefer my random forest model because it is very flexible compared to the other models. RF algorithms can effectively capture relationships and complex patterns within data by creating numerous decision trees, which can model non-linear boundaries and interactions between features. Specifically, with our mtry feature, at each node of a tree, the algorithm does not consider all available predictors. Instead, it randomly selects mtry number of predictors from the full set of predictors. This makes it less likely to select 2 highly correlated variables because one will be a more important predictor than the other (reducing the rmse more.) Unlike linear models that assume a straightforward relationship, the ensemble nature of a random forest allows it to learn intricate patterns that might be difficult for other algorithms to detect, making it a powerful and flexible tool for many machine learning tasks. The one thing that's not great about the RF is that it isn't interpretable. While individual trees are interpretable, the fact that we have a forest of trees with aggregate decision-making is difficult to follow. However, we can make plots and use techniques like feature importance to gain insights into the model's behavior and identify key drivers of its predictions.