Data-571 Assignment 2

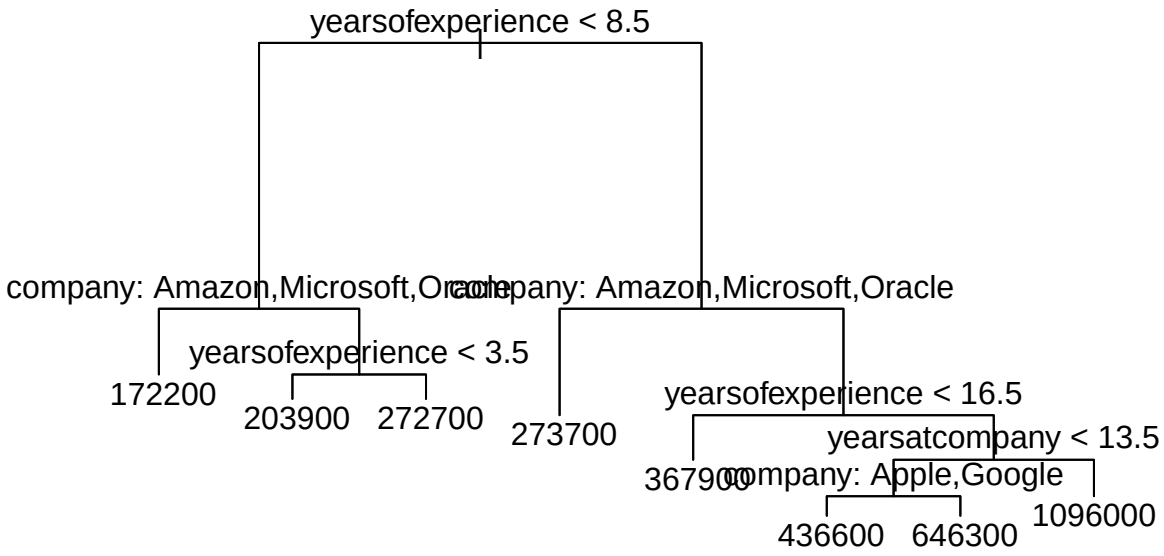
Dhun Sheth

2023-12-05

Question 1

Part A

```
mydata <- read.csv("datasalaries.csv", stringsAsFactors=TRUE)
tree_reg <- tree(totalyearlycompensation~., data = mydata)
plot(tree_reg)
text(tree_reg, pretty=0)
```



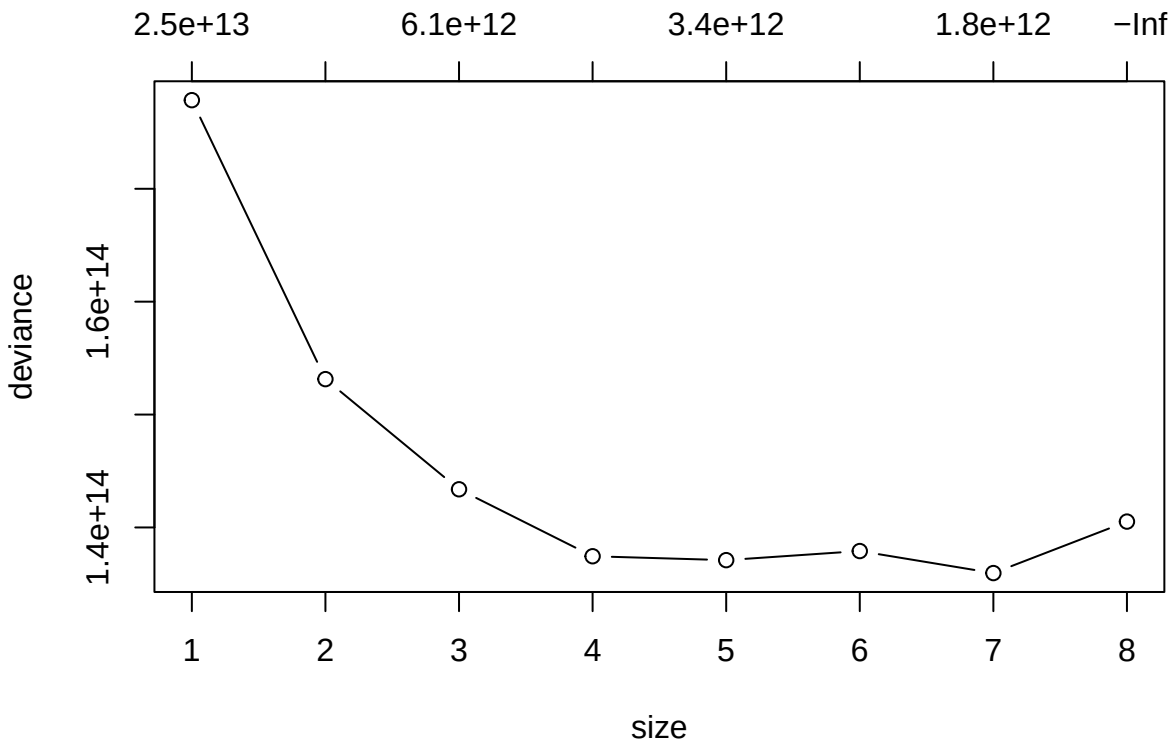
Part B

I would prefer to work for a company other than Amazon, Microsoft, Oracle, Apple or Google because this leads to a higher salary (646300) because assuming you are on the 2nd decision on the right side of the tree,

if you say no to working at Amazon, Microsoft, and Oracle -> this leads to a 5th possible decision where you if you work for Apple or Google you make 436600 but if you also don't work for Apple and Google you make around the 646300 amount.

Part C

```
set.seed(51341)
tree_cv <- cv.tree(tree_reg)
plot(tree_cv, type="b")
```



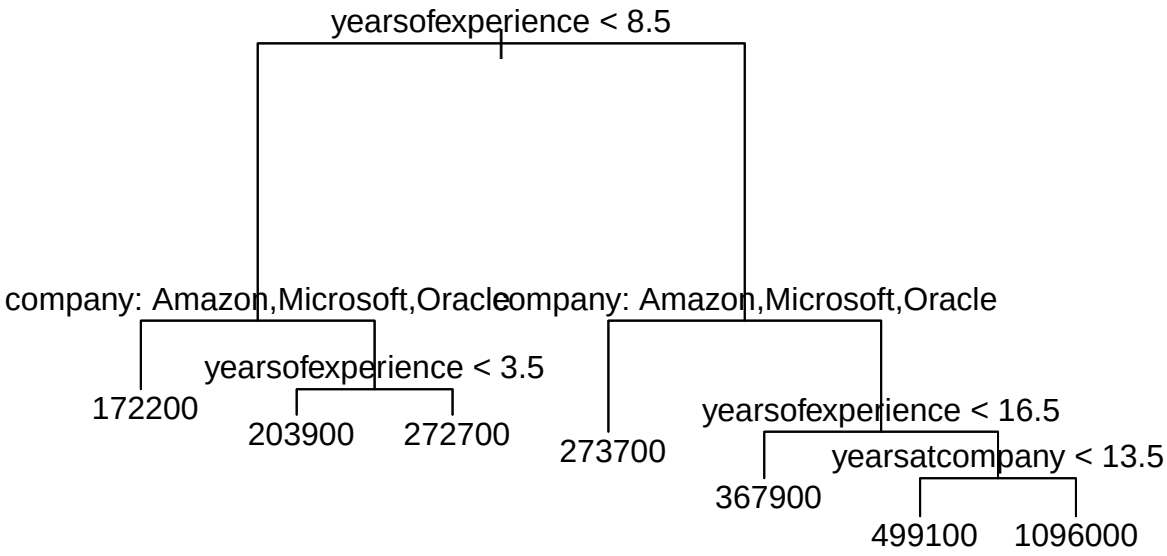

```
print(tree_cv$dev[2]/length(mydata$totalyearlycompensation))
```

```
## [1] 18087026057
```

10-fold CV suggests to use 7 terminal nodes.

Part D

```
pruned_tree_reg <- prune.tree(tree_reg, best=7)
plot(pruned_tree_reg)
text(pruned_tree_reg, pretty=0)
```




```
x0 <- data.frame(
  company = factor(c("Google"), levels=levels(mydata$company)),
  yearsofexperience = c(10),
  yearsatcompany = c(10),
  gender = factor(c("Female"), levels=levels(mydata$gender)),
  Race = factor(c("Asian"), levels=levels(mydata$Race)),
  Education = factor(c("PhD"), levels=levels(mydata$Education))
)

salary <- predict(pruned_tree_reg, x0)
print(salary)
```

```
##        1
## 367866.8
```

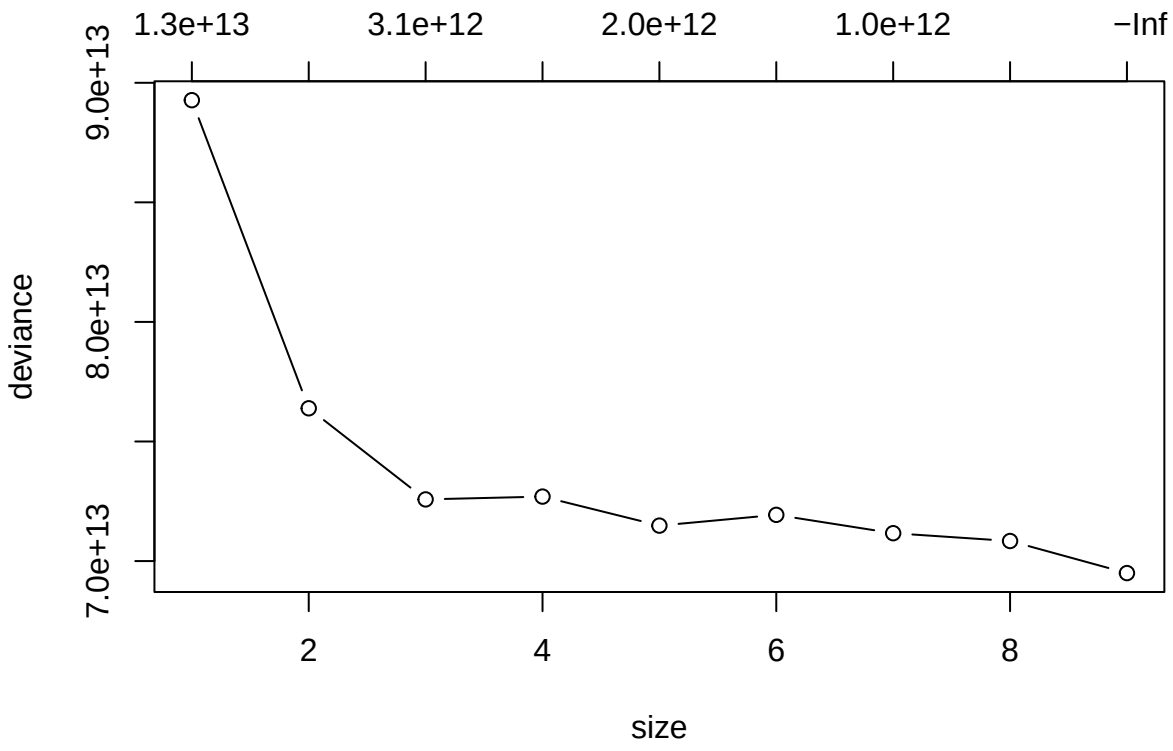
Predicted salary from the pruned tree is 3.6786685×10^5

Part E

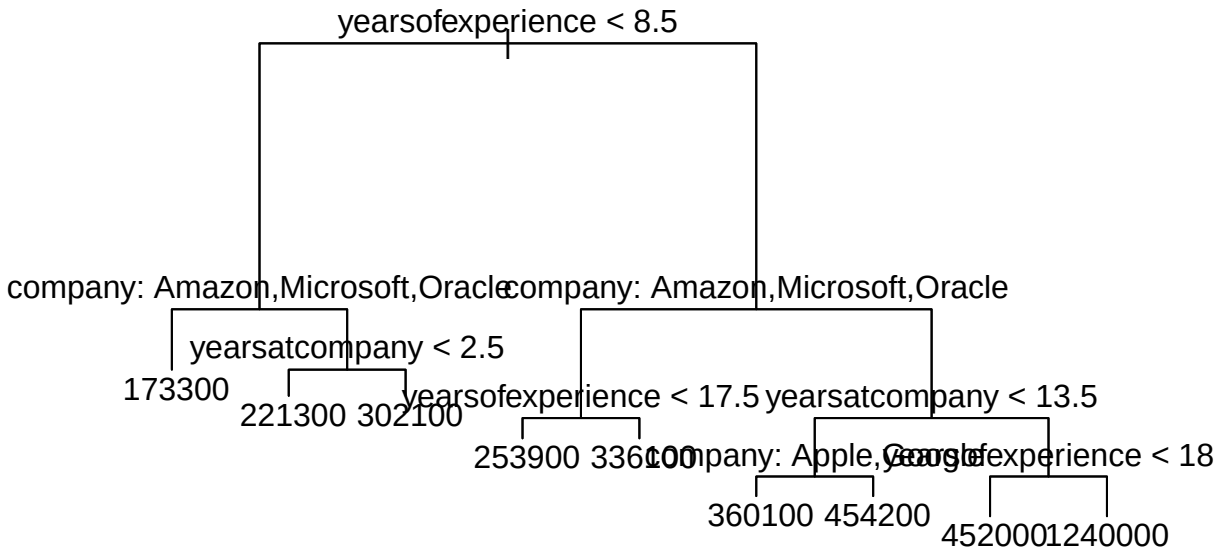
```
set.seed(763)
dsindex <- sample(1:nrow(mydata), 4000)
dstrain <- mydata[dsindex, ]
dstest <- mydata[-dsindex, ]
```

```r
tree_reg_train <- tree(totalyearlycompensation~., data = dstrain)

tree_train_cv <- cv.tree(tree_reg_train)
plot(tree_train_cv, type="b")
```




```r
plot(tree_reg_train)
text(tree_reg_train, pretty=0)
```




```
salary_train <- predict(tree_reg_train, x0)
print(salary_train)
```

```
##        1
## 360079.9
```

Based on CV, there is no need to prune the original tree made from the training set.

Predicted salary from the tree made on the training set is $\$3.6007987 \times 10^5$

Part F

```
RSS <- numeric(length(dstest$totalyearlycompensation))
for (i in 1:length(dstest$totalyearlycompensation)){
  RSS[i] <- (dstest[i,2] - predict(tree_reg_train, newdata=dstest[i,]))^2
}

est_MSE <- sum(RSS)/(length(RSS))
print(est_MSE)
```

```
## [1] 19324364849
```

Estimated MSE from test set is: 1.9324365×10^{10}
CV MSE estimate is: 1.8087026×10^{10}

The CV MSE estimate from part C is close to the test MSE.

Question 2

82.15451 is a more believable estimate for long-run MSE because during prediction, when the newdata argument isn't passed, the predict function will re-sample from the training data set to make a "new" dataset, however, if the newdata argument is passed, this tells the predict function it doesn't have to make a "new" dataset and uses the data directly. Therefore, in the last line it is calculating the MSE based on training data and is why the reported MSE is below the first 2 estimates (and is being under-estimated) because it used data the model was trained on, however, in the 2nd calculation of MSE where the newdata argument is passed, is a more accurate estimate of MSE.

Question 3

```
set.seed(4)
insurance_data <- read.csv("insurance.csv", stringsAsFactors=TRUE)

data_randomized <- sample(insurance_data)

d_train <- data_randomized[1:(2*length(data_randomized$charges)/3),]
d_test <- data_randomized[(2*length(data_randomized$charges)/3+1):(length(data_randomized$charges)),]
```

Linear Model

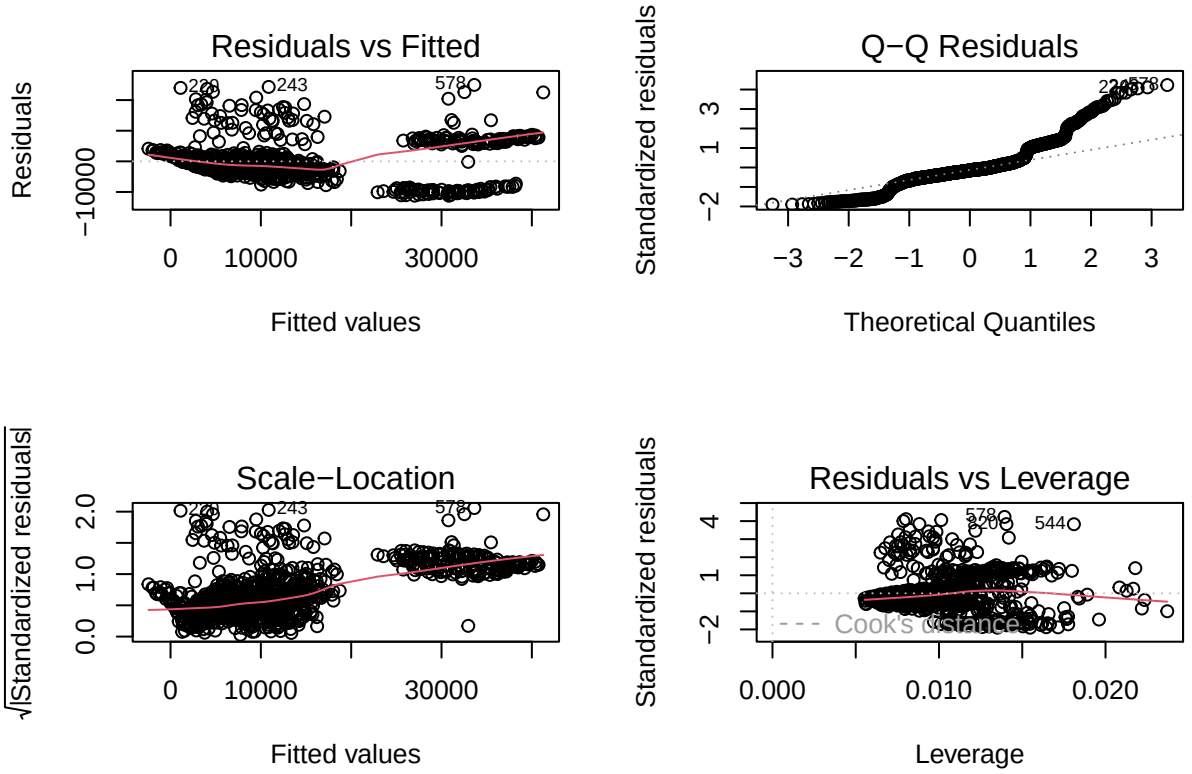
```
# set.seed(4)
# insurance_data <- read.csv("insurance.csv", stringsAsFactors=TRUE)
#
# data_randomized <- sample(insurance_data)
#
# d_train <- data_randomized[1:(2*length(data_randomized$charges)/3),]
# d_test <- data_randomized[(2*length(data_randomized$charges)/3+1):(length(data_randomized$charges)),]

lm_insu_reg <- lm(charges~., data=d_train)
summary(lm_insu_reg)
```

```
##
## Call:
## lm(formula = charges ~ ., data = d_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11216.1  -2806.9   -861.2   1307.5  24938.4
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -11968.80    1208.90  -9.901   <2e-16 ***
## bmi               344.80      34.54   9.984   <2e-16 ***
## regionnorthwest   -326.56     575.88  -0.567   0.5708
## regionsoutheast  -1083.18     564.02  -1.920   0.0551 .
## regionsouthwest   -919.89     571.83  -1.609   0.1080
## smokeryes        24110.11     501.28  48.097   <2e-16 ***
## children           400.41     167.66   2.388   0.0171 *
```

```
## sexmale           -595.33     399.69  -1.489   0.1367
## age                256.64      14.11  18.183   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5935 on 883 degrees of freedom
## Multiple R-squared:  0.7614, Adjusted R-squared:  0.7592
## F-statistic: 352.2 on 8 and 883 DF,  p-value: < 2.2e-16
```

```r
par(mfrow = c(2, 2))
plot(lm_insu_reg)
```




```r
par(mfrow = c(1, 1))

cvlm <- list()
msecv <- NA
for(i in 1:nrow(d_train)){
  cvlm[[i]] <- lm(charges~., data=d_train[-i,])
  msecv[i] <- (predict(cvlm[[i]], newdata=d_train[i,]) - d_train$charges[[i]])^2
}
lm_cv_MSE <- mean(msecv)
sprintf("CV MSE estimate for linear model on training data: %s", round(lm_cv_MSE, digits=3))
```

```
## [1] "CV MSE estimate for linear model on training data: 35678999.236"
```

```r
predicted_vals_test <- predict(lm_insu_reg, newdata = d_test)
lm_test_MSE <- mean((d_test$charges - predicted_vals_test)^2)
sprintf("MSE estimate for linear model on testing data: %s", round(lm_test_MSE, digits=3))
```

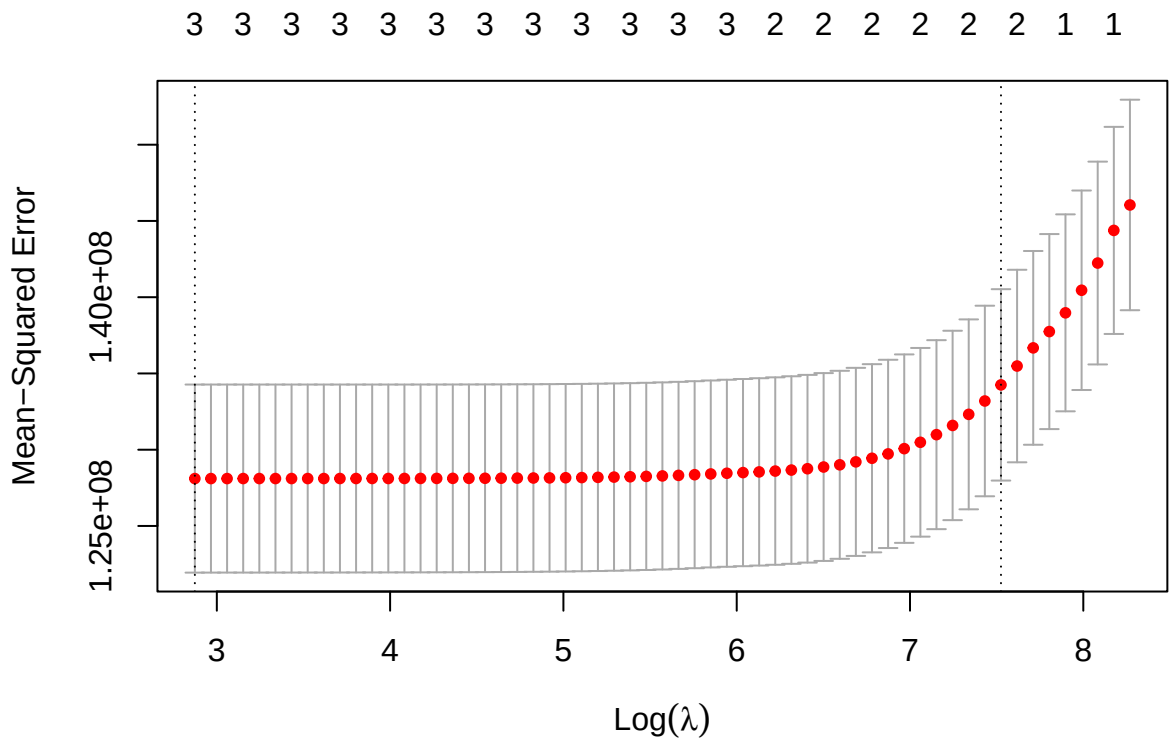
```
## [1] "MSE estimate for linear model on testing data: 40020943.282"
```

Based on the regression plots, the residuals don't seem independent or normally distributed or have constant variance which violate many of the assumptions necessary to fit a linear model.

Lasso

```r
x <- as.matrix(d_train[, -2])
y <- as.vector(d_train[, 2])


lasso_reg <- cv.glmnet(x,y, alpha = 1, lamda = lambda_values)
plot(lasso_reg)
```




```r
lasso_cv_mse <- min(lasso_reg$cvm)
print(lasso_cv_mse)
```

```
## [1] 128107020
```

```r
best_lasso_reg <- glmnet(x, y, alpha = 1, lambda = lasso_reg$lambda.min)

predicted_vals_test <- predict(best_lasso_reg, newx = as.matrix(d_test[,-2]))
lasso_test_MSE <- mean((d_test$charges - predicted_vals_test)^2)
sprintf("MSE estimate for Lasso regression on testing data: %s", round(lasso_test_MSE, digits=3))
```

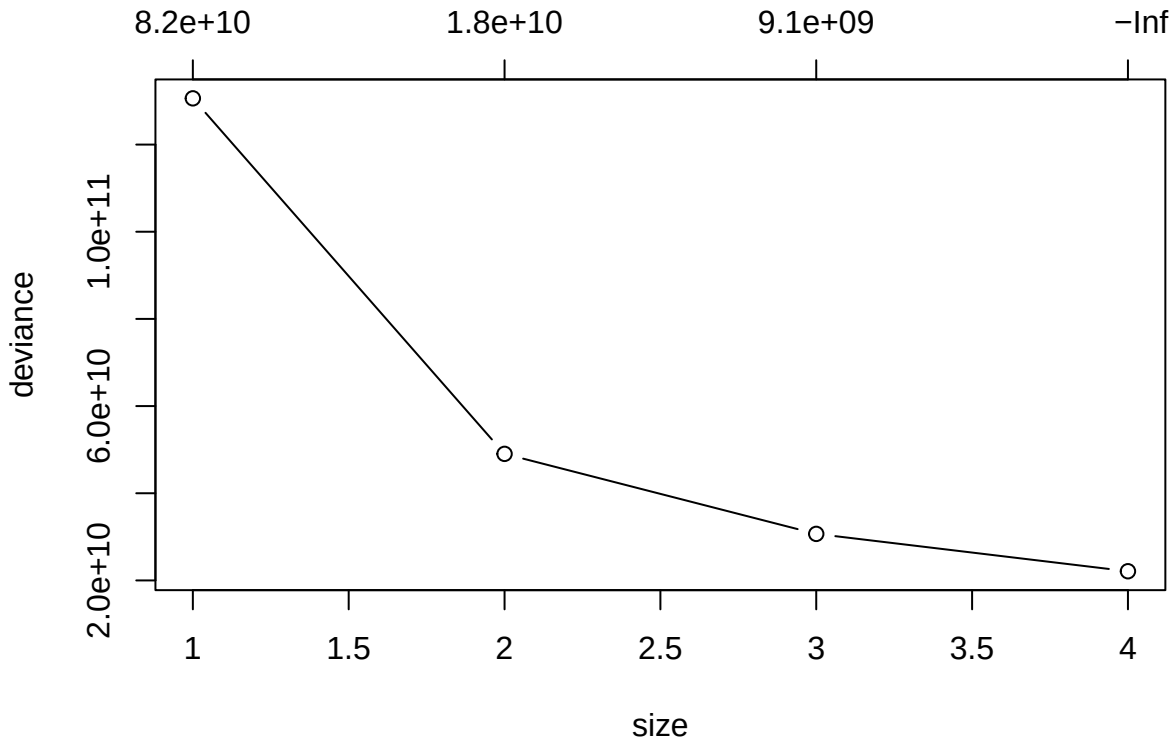
```
## [1] "MSE estimate for Lasso regression on testing data: 133521776.277"
```

Trees

```r
# set.seed(4)
# insurance_data <- read.csv("insurance.csv", stringsAsFactors=TRUE)
#
# data_randomized <- sample(insurance_data)
#
# d_train <- data_randomized[1:(2*length(data_randomized$charges)/3),]
# d_test <- data_randomized[(2*length(data_randomized$charges)/3+1):(length(data_randomized$charges)),]

tree_insu_reg <- tree(charges~., data = d_train)

set.seed(51341)
tree_insu_cv <- cv.tree(tree_insu_reg)
plot(tree_insu_cv, type="b")
```



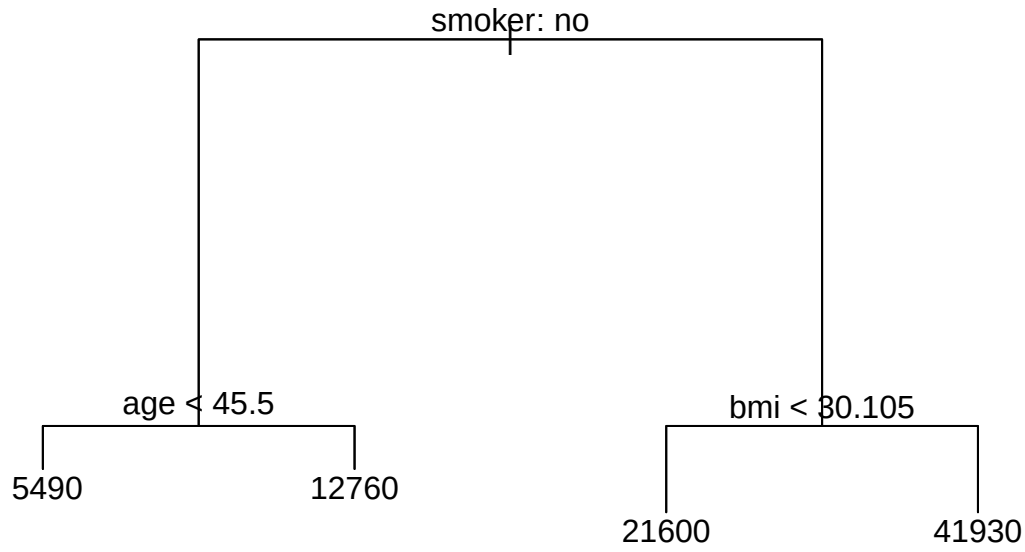
```r
print("No pruning needed")
```

```
## [1] "No pruning needed"
```

```r
tree_cv_MSE <- min(tree_insu_cv$dev)/length(insurance_data$charges)
sprintf("Tree CV MSE estimate based on training data: %s", round(tree_cv_MSE,digits=3)) # Tree CV MSE E
```

```
## [1] "Tree CV MSE estimate based on training data: 16522123.162"
```

```r
plot(tree_insu_reg)
text(tree_insu_reg, pretty=0)
```




```r
summary(tree_insu_reg)
```

```
##
## Regression tree:
## tree(formula = charges ~ ., data = d_train)
## Variables actually used in tree construction:
## [1] "smoker" "age"    "bmi"
## Number of terminal nodes:  4
## Residual mean deviance:  24120000 = 2.142e+10 / 888
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -9381   -3101   -1037       0    1202   22990
```

```r
predicted_vals_test <- predict(tree_insu_reg, newdata = d_test)
tree_test_MSE <- mean((d_test$charges - predicted_vals_test)^2)

sprintf("Tree MSE estimate based on testing data: %s", round(tree_test_MSE,digits=3))
```

```
## [1] "Tree MSE estimate based on testing data: 28823125.572"
```

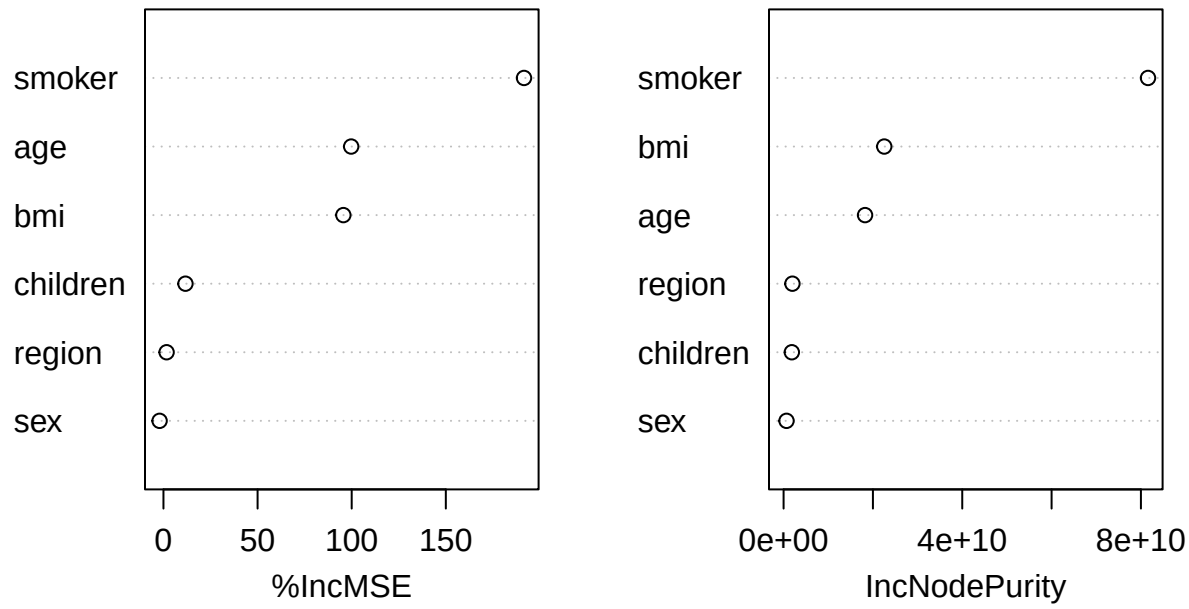
Random Forest

```r
# set.seed(4)
# insurance_data <- read.csv("insurance.csv", stringsAsFactors=TRUE)
#
# data_randomized <- sample(insurance_data)
#
# d_train <- data_randomized[1:(2*length(data_randomized$charges)/3),]
# d_test <- data_randomized[(2*length(data_randomized$charges)/3+1):(length(data_randomized$charges)),]

RF_insu_reg <- randomForest(charges~., data=d_train, mtry=4, importance=TRUE)
print(RF_insu_reg)
```

```
##
## Call:
##  randomForest(formula = charges ~ ., data = d_train, mtry = 4,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 21319469
##                     % Var explained: 85.41
```

```r
varImpPlot(RF_insu_reg)
```




```
RF_cv_MSE <- RF_insu_reg$mse[500]
sprintf("Random forest MSE estimate based on OOB estimates from training data: %s",RF_cv_MSE)
```

```
## [1] "Random forest MSE estimate based on OOB estimates from training data: 21319469.2140666"
```

```
predicted_vals_test <- predict(RF_insu_reg, newdata = d_test)
RF_test_MSE <- mean((d_test$charges - predicted_vals_test)^2)

sprintf("Tree MSE estimate based on testing data: %s", round(RF_test_MSE,digits=3))
```

```
## [1] "Tree MSE estimate based on testing data: 23890949.266"
```

Boosting

```
# set.seed(4)
# insurance_data <- read.csv("insurance.csv", stringsAsFactors=TRUE)
#
# data_randomized <- sample(insurance_data)
#
# d_train <- data_randomized[1:(2*length(data_randomized$charges)/3),]
# d_test <- data_randomized[(2*length(data_randomized$charges)/3+1):(length(data_randomized$charges)),]
```

```r
boost_insu_reg <- gbm(charges~., distribution="gaussian", data=d_train, n.trees=5000, interaction.depth=

# Create a training control object for cross-validation
ctrl <- trainControl(method = "cv", number = 20)  # 5-fold cross-validation

# Use the train function from caret for cross-validation
cv_results <- train(charges ~ ., data = d_train, method = "gbm", trControl = ctrl, verbose = FALSE)

# Access the cross-validated error values
boost_cv_MSE <- mean(cv_results$results$RMSE^2)
sprintf("Boosted CV MSE estimate based on training data: %s",boost_cv_MSE)
```

```
## [1] "Boosted CV MSE estimate based on training data: 24635257.3156289"
```

```r
predicted_vals_test <- predict(boost_insu_reg, newdata = d_test)
```

```
## Using 5000 trees...
```

```r
boost_test_MSE <- mean((d_test$charges - predicted_vals_test)^2)
sprintf("Boosted MSE estimate based on testing data: %s",boost_test_MSE)
```

```
## [1] "Boosted MSE estimate based on testing data: 22739702.4477322"
```

Model	CV MSE	Test MSE
Linear Model	3.5678999×10^7	4.0020943×10^7
Lasso	1.2810702×10^8	1.3352178×10^8
Trees	1.6522123×10^7	2.8823126×10^7
Random Forest	2.1319469×10^7	2.3890949×10^7
Boosting	2.4635257×10^7	2.2739702×10^7

Based on the above metrics for test MSE, the best model to choose based on sole prediction performance is the Boosted model because it gives the lowest test MSE however the Random Forest has the lowest CV MSE estimate.

If I was consulting for an insurance company, I would choose the tree model because it's easier to interpret and the performance is not that much worse than the boosted model. A tree is very dependent on the training data sample and so we expect it to perform worse when compared to other models such as the random forest or boosted model.