Week 8 Exercise Solutions

Question 1

(a)

Q: Let's begin by building a logistic regression model to predict whether an individual's earnings are above $50,000 (the variable "over50k") using all of the other variables as independent variables. Split the data randomly into a training set and a testing set, setting the seed to 2000 before creating the split. Split the data so that the training set contains 60% of the observations, while the testing set contains 40% of the observations. Next, build a logistic regression model to predict the dependent variable "over50k", using all of the other variables in the dataset as independent variables. Use the training set to build the model. Identify all the variables that are significant, or have factors that are significant? (Use 0.1 as your significance threshold. You might see a warning message here - you can ignore it and proceed. This message is a warning that we might be overfitting our model to the training set.)

A:

```
df_1 <- read.csv("census.csv")
# str(df_1)
library(caTools)
set.seed(2000)
spl_1 <- sample.split(df_1$over50k, SplitRatio = 0.6)
train_1 <- subset(df_1, spl_1 == TRUE)
test_1 <- subset(df_1, spl_1 == FALSE)
glm_1 <- glm(over50k ~ ., family = "binomial", data = train_1)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

The follow code will print out all predictors significant at the 10% significance level.

```
names(which(coef(summary(glm_1))[,4] < 0.1))
```

```
##  [1] "(Intercept)"                       "age"
##  [3] "workclass Federal-gov"             "workclass Local-gov"
##  [5] "workclass Private"                 "workclass Self-emp-inc"
##  [7] "workclass State-gov"               "education 12th"
##  [9] "education Assoc-acdm"              "education Assoc-voc"
## [11] "education Bachelors"               "education Doctorate"
## [13] "education HS-grad"                 "education Masters"
## [15] "education Prof-school"             "education Some-college"
## [17] "maritalstatus Married-AF-spouse"   "maritalstatus Married-civ-spouse"
## [19] "maritalstatus Never-married"       "occupation Craft-repair"
## [21] "occupation Exec-managerial"        "occupation Farming-fishing"
## [23] "occupation Handlers-cleaners"      "occupation Other-service"
## [25] "occupation Prof-specialty"         "occupation Protective-serv"
## [27] "occupation Sales"                  "occupation Tech-support"
## [29] "relationship Not-in-family"        "relationship Own-child"
## [31] "relationship Unmarried"            "relationship Wife"
## [33] "sex Male"                          "capitalgain"
## [35] "capitalloss"                       "hoursperweek"
```

We list the significant variables below:

- age
- workclass
  - Federal-gov, Local-gov, Private, Self-emp-inc, State-gov
- education
  - 12th, Assoc-acdm, Assoc-voc, Bachelors, Doctorate, HS-grad, Masters, Prof-school, Some-college
- maritalstatus
  - Married-AF-spouse, Married-civ-spouse, Never-married
- occupation
  - Craft-repair, Exec-managerial, Farming-fishing, Handlers-cleaners, Other-service, Prof-specialty, Protective-serv, Sales, Tech-support
- relationship
  - Not-in-family, Own-child, Unmarried, Wife

- sex
    - Male
- capitalgain
- capitalloss
- hoursperweek

(b)

Q: What is the accuracy of the model on the testing set? Use a threshold of 0.5. (You might see a warning message when you make predictions on the test set - you can safely ignore it.)

A:

```
pred_1_b <- predict(glm_1, newdata = test_1, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
## == : prediction from a rank-deficient fit may be misleading
```

```
table_1_b <- table(test_1$over50k, pred_1_b >= 0.5)
sum(diag(table_1_b))/sum(table_1_b)
```

```
## [1] 0.8552107
```

The accuracy of the model on the testing set is 0.8552107.

(c)

Q: What is the baseline accuracy for the testing set?

A: We need to get the baseline or more common category from the base set:

```
base_1 <- names(table(train_1$over50k)[which.max(table(train_1$over50k))])
unname(table(test_1$over50k)[base_1]/sum(nrow(test_1)))
```

```
## [1] 0.7593621
```

The baseline accuracy for the testing set is 0.7593621.

(d)

Q: What is the area-under-the-curve (AUC) for this model on the test set?

A:

```
library(ROCR)
rocr_1_d <- prediction(pred_1_b, test_1$over50k)
performance(rocr_1_d, "auc")@y.values
```

```
## [[1]]
## [1] 0.9061598
```

The AUC for this model on the test set is 0.906159757757142.

(e)

Q: We have just seen how the logistic regression model for this data achieves a high accuracy. Moreover, the significances of the variables give us a way to gauge which variables are relevant for this prediction task. However, it is not immediately clear which variables are more important than the others, especially due to the large number of factor variables in this problem. Let us now build a classification tree to predict "over50k". Use the training set to build the model, and all of the other variables as independent variables. Use the default parameters. After you are done building the model, plot the resulting tree.

A: First, we load the library for building a classification tree:

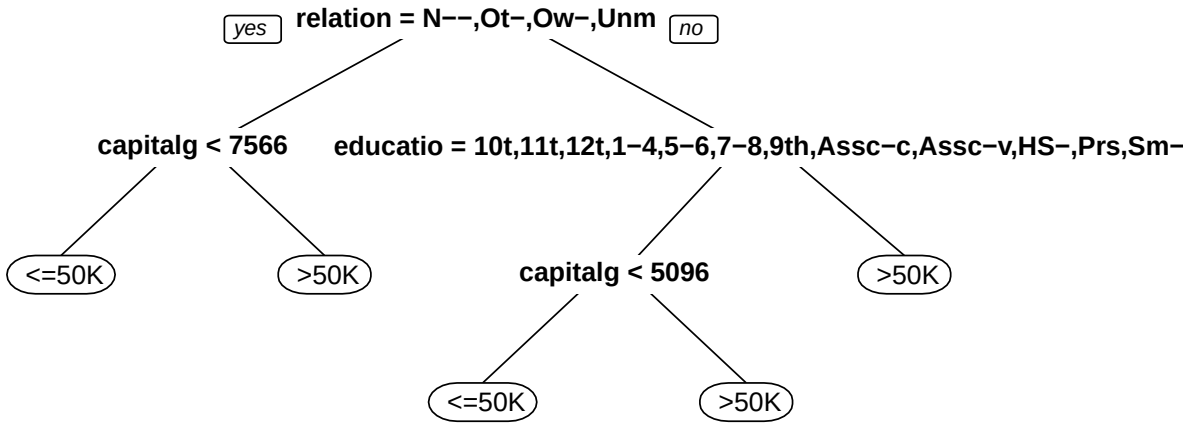
```
library(rpart)
library(rpart.plot)
```

Then we build the tree

```
tree_1_e <- rpart(over50k ~ ., data = train_1, method = "class")
```

And plot it:

```
prp(tree_1_e)
```



(f)

Q: How many splits does the tree have in total?

```
unname(tail(tree_1_e$cptable[,2], 1))
```

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

A: There are 4 splits in the tree, corresponding to the default `cp` parameter of `0.01`.

(g)

Q: Which variable does the tree split on at the first level (the very first split of the tree)?

A: We can see this in the `prp()` output above, or use the following

```
# row name is the node index, in this case 1 for the first split
# and first column is the variable
tree_1_e$frame["1", 1]
```

```
## [1] relationship
## Levels: <leaf> capitalgain education relationship
```

The tree splits on the variable `relationship` at the first level.

(h)

Q: Which variables does the tree split on at the second level (immediately after the first split of the tree)?

A:

```
# row name is the node index, in this case 1 for the first split
# and first column is the variable
tree_1_e$frame["2", 1]
tree_1_e$frame["3", 1]
```

```
## [1] capitalgain
## Levels: <leaf> capitalgain education relationship
## [1] education
## Levels: <leaf> capitalgain education relationship
```

The tree splits on the variables `capitalgain` and `education` at the second level.

(i)

Q: What is the accuracy of the model on the testing set? Use a threshold of 0.5.

A:

```
pred_1_i <- predict(tree_1_e, newdata = test_1, type = "class")
table_1_i <- table(test_1$over50k, pred_1_i)
sum(diag(table_1_i))/sum(table_1_i)
```

```
## [1] 0.8473927
```

The accuracy of the model on the testing is 0.8473927.

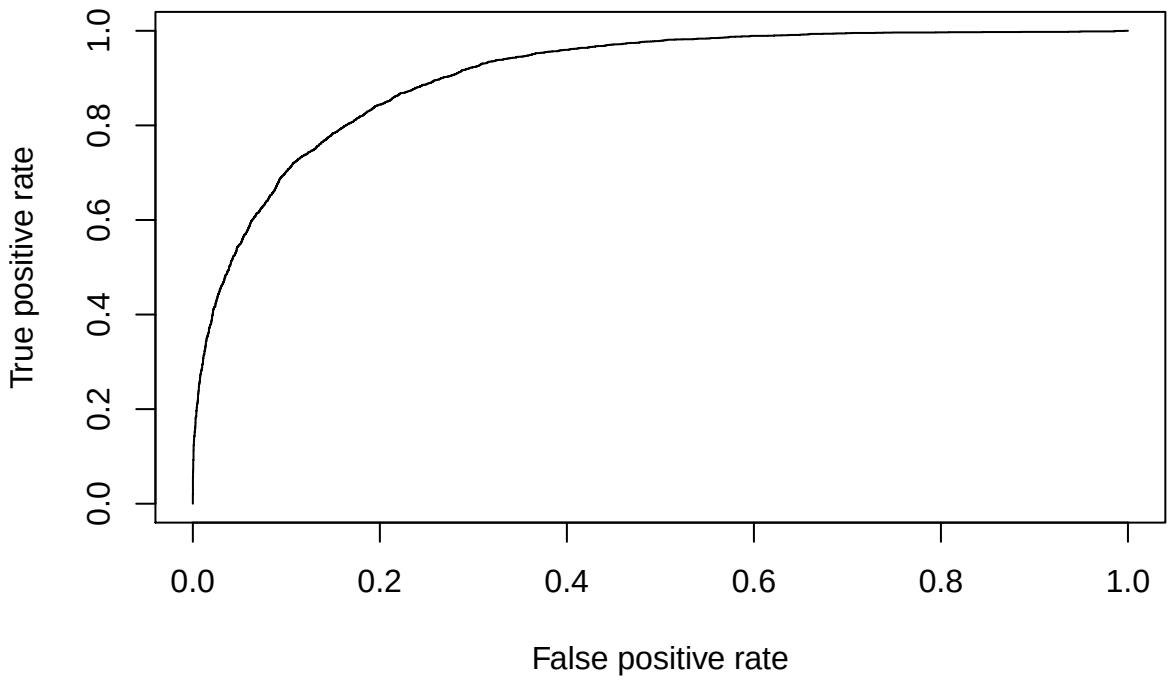
(j)

Q: Let us now consider the ROC curve and AUC for the CART model on the test set. You will need to get predicted probabilities for the observations in the test set to build the ROC curve and compute the AUC. Plot the ROC curve for the CART model you have estimated. Observe that compared to the logistic regression ROC curve, the CART ROC curve is less smooth. Which of the following explanations for this behavior is most correct?

- The number of variables that the logistic regression model is based on is larger than the number of variables used by the CART model, so the ROC curve for the logistic regression model will be smoother.
- CART models require a higher number of observations in the testing set to produce a smoother/more continuous ROC curve; there is simply not enough data.
- The probabilities from the CART model take only a handful of values (five, one for each end bucket/leaf of the tree); the changes in the ROC curve correspond to setting the threshold to one of those values.
- The CART model uses fewer continuous variables than the logistic regression model (`capitalgain` for CART versus `age`, `capitalgain`, `capitallosses`, `hoursperweek`), which is why the CART ROC curve is less smooth than the logistic regression one.

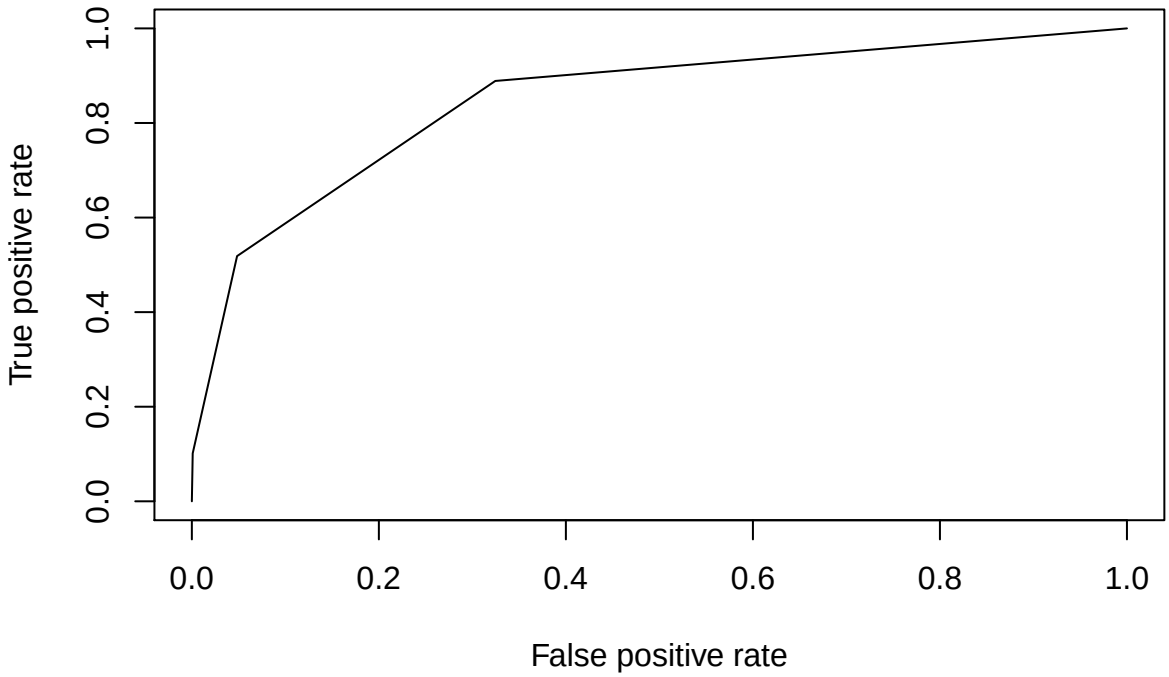
A: We first plot the ROC curve for the logistic regression model:

```
perf_1_j_1 <- performance(rocr_1_d, measure = "tpr", x.measure = "fpr")
plot(perf_1_j_1)
```



Then we plot the ROC curve for the CART model:

```
pred_1_j <- predict(tree_1_e, newdata = test_1, type = "prob")
rocr_1_j <- prediction(pred_1_j[,2], test_1$over50k)
perf_1_j_2 <- performance(rocr_1_j, measure = "tpr", x.measure = "fpr")
plot(perf_1_j_2)
```



The plots indicate that the ROC curve for logistic regression is more smooth. By tabulating the probability results (as opposed to majority votes) from the CART model, we can see that there are only 5 possible predicted probability values. Since they can only take on these handful of values, the breakpoints of the curve correspond to the true positive rate and false positive rate when the threshold is set to these values. Option 3 is correct.

(k)

Q: What is the AUC of the CART model on the test set?

A:

```
performance(rocr_1_j, measure = "auc")@y.values
```

```
## [[1]]
## [1] 0.8470256
```

The AUC of the CART model on the test set is 0.847025569517672.

(l)

Q: Before building a random forest model, we will down-sample our training set. While some modern personal computers can build a random forest model on the entire training set, others might run out of memory when trying to train the model since random

forests is much more computationally intensive than CART or Logistic Regression. For this reason, before continuing we will define a new training set to be used when building our random forest model, that contains 2000 randomly selected observations from the original training set. Do this by running the following commands in your R console (assuming your training set is called "train"):

```
> set.seed(1)
> trainSmall <- train[sample(nrow(train), 2000), ]
```

Let us now build a random forest model to predict "over50k", using the dataset "trainSmall" to build the model. Set the seed to 1 again right before building the model, and use all of the other variables in the dataset as independent variables. (If you get an error that random forest "can not handle categorical predictors with more than 32 categories", rebuild the model without the `nativecountry` variable as one of the independent variables.) Then, make predictions using this model on the entire test set. What is the accuracy of the model on the test set, using a threshold of 0.5? (Remember that you don't need a "type" argument when making predictions with a random forest model if you want to use a threshold of 0.5. Also, note that your accuracy might be different from since the random forest models can still differ depending on your operating system, even when the random seed is set.)

A: Subsetting,

```
set.seed(1)
trainSmall <- train_1[sample(nrow(train_1), 2000),]
```

Then running the random forest algorithm,

```
library(randomForest)
set.seed(1)
rf_1 <- randomForest(over50k ~ ., data = trainSmall)
pred_1_l <- predict(rf_1, newdata = test_1)
table_1_l <- table(test_1$over50k, pred_1_l)
sum(diag(table_1_l))/sum(table_1_l)
```

```
## [1] 0.8527871
```

(m)

Q: As we discussed in class, random forest models work by building a large collection of trees. As a result, we lose some of the interpretability that comes with CART in terms of seeing how predictions are made and which variables are important. However, we can still compute metrics that give us insight into which variables are important. One

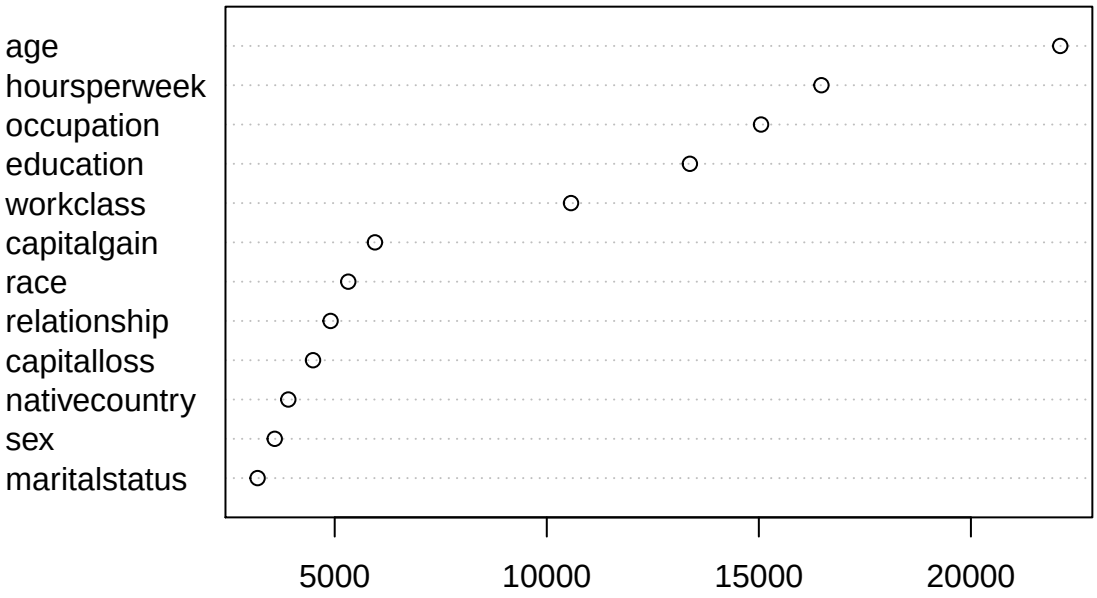
metric that we can look at is the number of times, aggregated over all of the trees in the random forest model, that a certain variable is selected for a split. To view this metric, run the following lines of R code (replace "MODEL" with the name of your random forest model):

```
> vu <- varUsed(MODEL, count=TRUE)
> vusorted <- sort(vu, decreasing = FALSE, index.return = TRUE)
> dotchart(vusorted$x, names(MODEL$forest$xlevels[vusorted$ix]))
```

This code produces a chart that for each variable measures the number of times that variable was selected for splitting (the value on the x-axis). Which of the variables is the most important in terms of the number of splits?

A:

```
vu <- varUsed(rf_1, count = TRUE)
vusorted <- sort(vu, decreasing = FALSE, index.return = TRUE)
dotchart(vusorted$x, names(rf_1$forest$xlevel[vusorted$ix]))
```



We can get the most frequent variable by calling `tail()` on the last `names()` object:

```
tail(names(rf_1$forest$xlevel[vusorted$ix]), 1)
```

```
## [1] "age"
```

The variable that is the most important in terms of the number of splits is `age`.

(n)

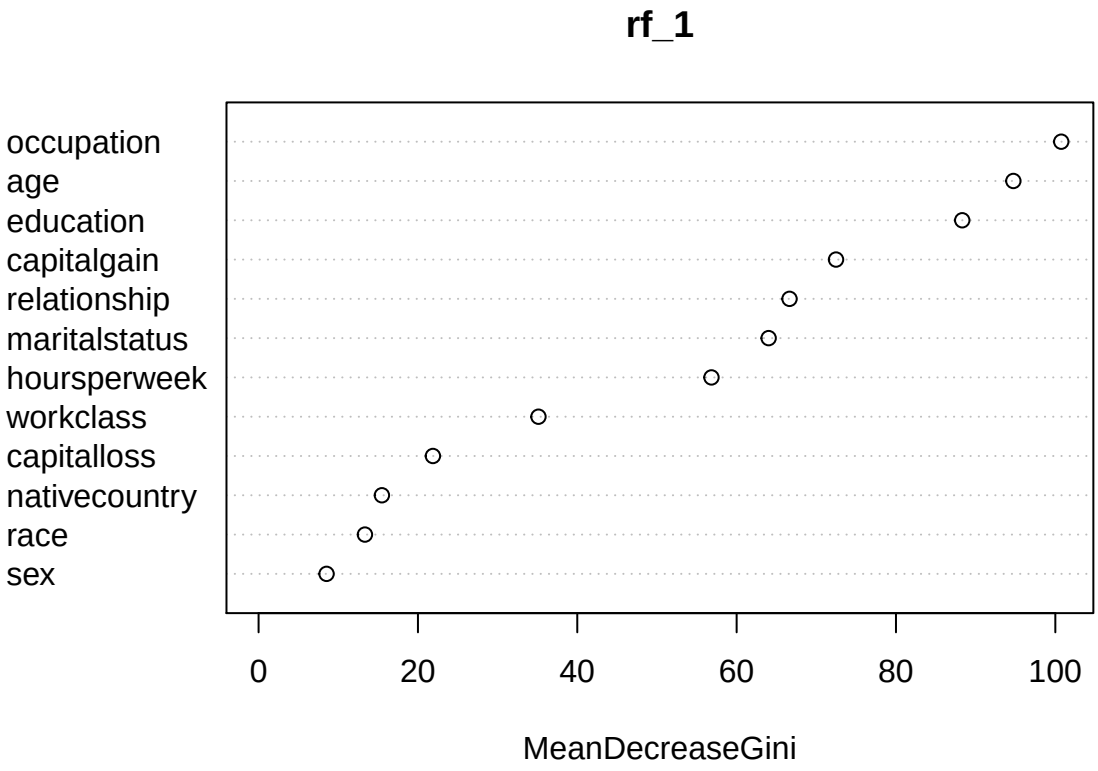
Q: A different metric we can look at is related to “impurity”, which measures how homogeneous each bucket or leaf of the tree is. In each tree in the forest, whenever we select a variable and perform a split, the impurity is decreased. Therefore, one way to measure the importance of a variable is to average the reduction in impurity, taken over all the times that variable is selected for splitting in all of the trees in the forest. To compute this metric, run the following command in R (replace ”MODEL” with the name of your random forest model):

```
> varImpPlot(MODEL)
```

Which of the following variables is the most important in terms of mean reduction in impurity?

A: We can plot as directed by the question,

```
varImpPlot(rf_1)
```



or grab it directly from the object:

```
rownames(rf_1$importance)[which.max(rf_1$importance)]
```

```
## [1] "occupation"
```

The variable `occupation` is the most important in terms of mean reduction in impurity.

(o)

Q: We now conclude our study of this dataset by looking at how CART behaves with different choices of its parameters. Let us select the cost complexity parameter for our CART model using k-fold cross validation, with k = 10 folds. Modify the minimum complexity parameter cp = 0.0001. Suggest a reasonable value of the cost complexity parameter from the plot and plot the corresponding tree.

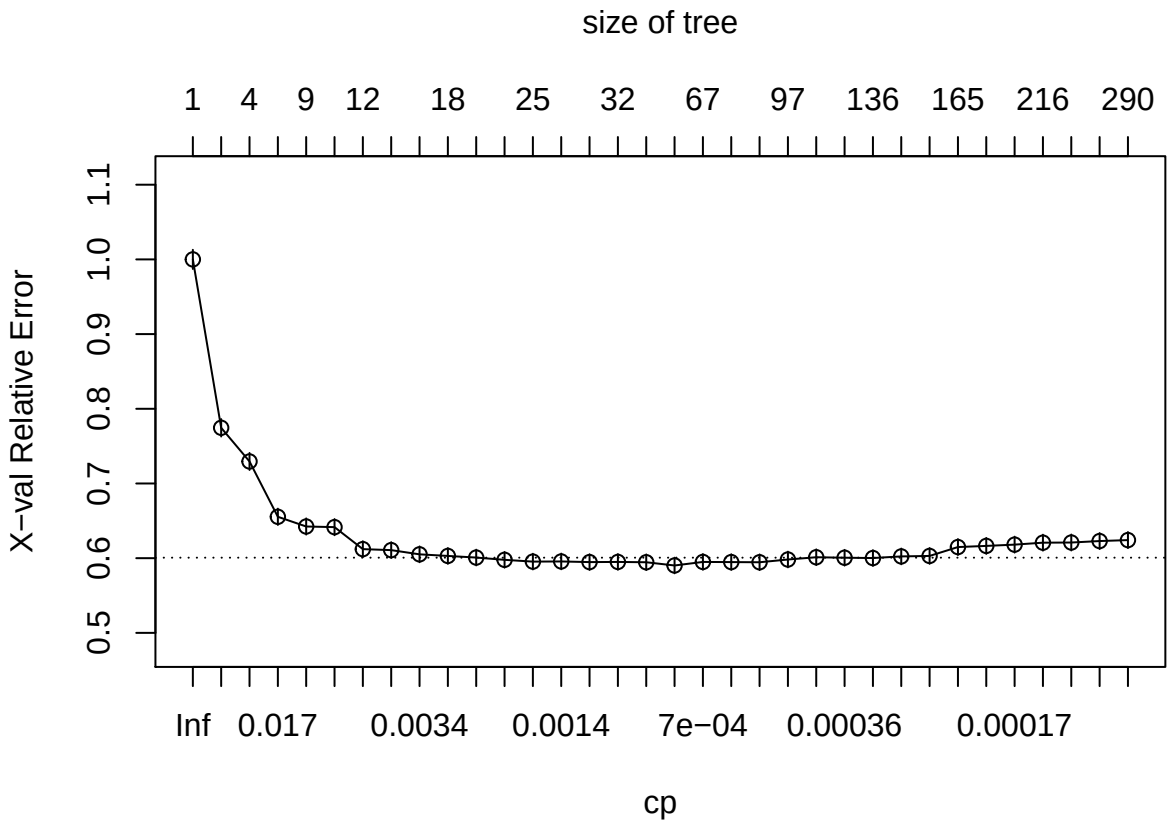
A: We run CART with specified `cp` parameter below:

```
tree_1_o_1 <- rpart(over50k ~., data = train_1, cp = 0.0001)
printcp(tree_1_o_1)  # uncomment but output is fairly big and uninformative
```

```
##
## Classification tree:
## rpart(formula = over50k ~ ., data = train_1, cp = 1e-04)
##
## Variables actually used in tree construction:
##  [1] age           capitalgain   capitalloss   education     hoursperweek
##  [6] maritalstatus nativecountry occupation    race          relationship
## [11] sex           workclass
##
## Root node error: 4617/19187 = 0.24063
##
## n= 19187
##
##            CP nsplit rel error  xerror     xstd
## 1  0.12183236      0   1.00000 1.00000 0.012825
## 2  0.06562703      2   0.75634 0.77453 0.011683
## 3  0.03747022      3   0.69071 0.72948 0.011413
## 4  0.00758068      4   0.65324 0.65540 0.010935
## 5  0.00595625      8   0.62292 0.64241 0.010846
## 6  0.00433182     10   0.61100 0.64154 0.010840
## 7  0.00422352     11   0.60667 0.61209 0.010632
## 8  0.00346545     13   0.59822 0.61079 0.010623
## 9  0.00324886     16   0.58696 0.60515 0.010582
## 10 0.00216591     17   0.58371 0.60299 0.010567
## 11 0.00194932     18   0.58155 0.60082 0.010551
## 12 0.00151614     19   0.57960 0.59779 0.010529
## 13 0.00144394     24   0.57050 0.59541 0.010511
## 14 0.00129955     28   0.56444 0.59562 0.010513
## 15 0.00108295     29   0.56314 0.59476 0.010506
## 16 0.00103964     31   0.56097 0.59498 0.010508
## 17 0.00086636     39   0.55187 0.59454 0.010505
## 18 0.00075807     62   0.52697 0.59021 0.010473
```

```
## 19 0.00064977     66   0.52393 0.59498 0.010508
## 20 0.00054148     70   0.52090 0.59476 0.010506
## 21 0.00051982     80   0.51527 0.59454 0.010505
## 22 0.00043318     96   0.50162 0.59822 0.010532
## 23 0.00039708    109   0.49556 0.60126 0.010554
## 24 0.00032489    116   0.49274 0.60061 0.010549
## 25 0.00030323    135   0.48625 0.60017 0.010546
## 26 0.00028879    140   0.48473 0.60234 0.010562
## 27 0.00025991    154   0.48018 0.60299 0.010567
## 28 0.00021659    164   0.47693 0.61490 0.010653
## 29 0.00017327    191   0.47109 0.61642 0.010663
## 30 0.00016244    205   0.46827 0.61815 0.010676
## 31 0.00015471    215   0.46524 0.62075 0.010694
## 32 0.00014439    242   0.45896 0.62097 0.010696
## 33 0.00010830    255   0.45701 0.62292 0.010710
## 34 0.00010000    289   0.45332 0.62421 0.010719
```

```
plotcp(tree_1_o_1)
```



If we were to strictly follow the minimum cross-validation error, we could run the following:

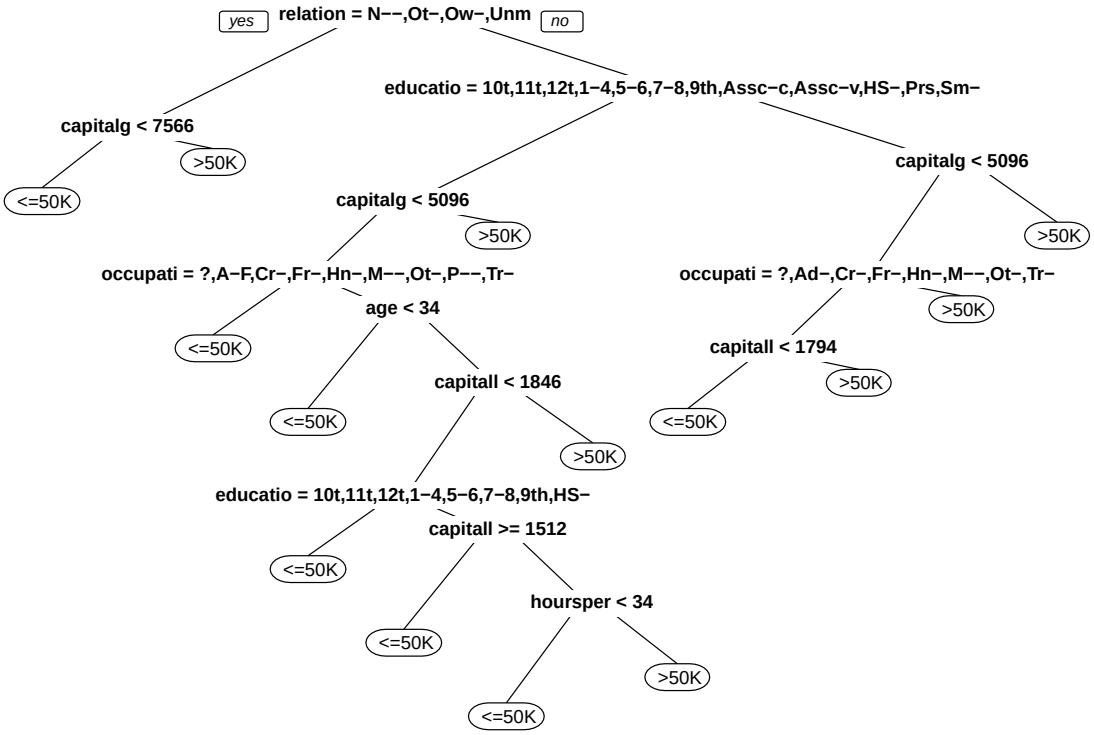
```
tree_1_o_1$cptable[which.min(tree_1_o_1$cptable[,4]),]
```

```
##           CP       nsplit    rel error       xerror         xstd
##  0.000758068 62.000000000  0.526965562  0.590210093  0.010472760
```

which suggests to prune to the `cp` parameter of 0.000758068.

However, looking at the table, we can see that many of the splits result in an `xerror` of around 0.61, 0.60 and 0.59. Any value which results in a similar cross-validation error would work. We aim for the smallest tree with around `xerror` of 0.61, the first of which has `cp` value `0.00422352`.

```
tree_1_o_2 <- prune(tree_1_o_1, cp = 0.00422352)
prp(tree_1_o_2)
```



(p)

Q: What is the prediction accuracy on the test set? Comment on how the model compares with the model in part (e).

A:

```
pred_1_p <- predict(tree_1_o_2, newdata = test_1, type = "class")
table_1_p <- table(test_1$over50k, pred_1_p)
table_1_p
```

```
##         pred_1_p
##           <=50K  >50K
##    <=50K   9145   568
##    >50K    1256  1822
```

```
sum(diag(table_1_p))/sum(table_1_p)
```

```
## [1] 0.8573997
```

The accuracy on the test set is 0.8573997.

The accuracy on the test set is greater but the model is much more complicated and less interpretable.

Question 2

(a)

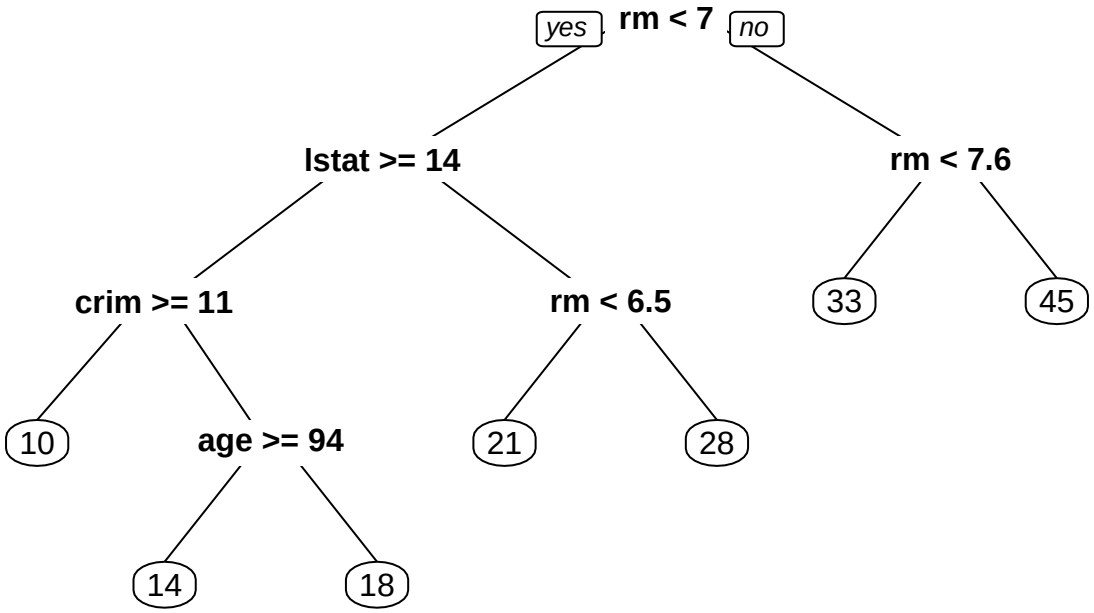
Q: Use a seed of 1. Split the dataset into a training set and a test set using the sample function where half the observations lie in each set. Fit a regression tree to the training set. Plot the tree. How many predictor variables were used in the regression tree?

A: Splitting the dataset,

```
df_2 <- read.csv("Boston.csv")
set.seed(1)
trainid_2 <- sample(1:nrow(df_2), nrow(df_2)/2)
train_2 <- df_2[trainid_2,]
test_2 <- df_2[-trainid_2,]
```

Running the CART model and plotting,

```
library(rpart)
library(rpart.plot)
tree_2_a <- rpart(medv ~ ., data = train_2)
# summary(tree_2_a)
# tree_2_a
prp(tree_2_a)
```



The variables used to split are `rm`, `lstat`, `crim` and `age`.

(b)

Q: What is the test set mean squared error? Draw a scatter plot of the fitted and true values. On average, the test predictions are within what range of the true median home value for the suburb?

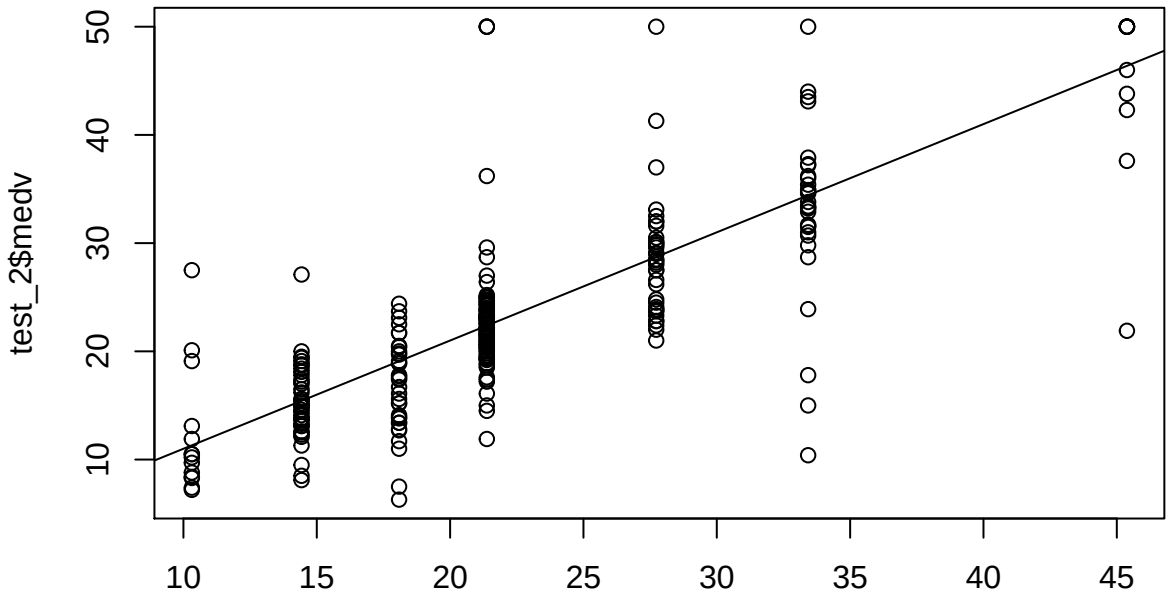
A: Calculating the mean squared error,

```
pred_2 <- predict(tree_2_a, newdata = test_2)
mse_2_b <- mean((pred_2 - test_2$medv)^2)
mse_2_b
```

```
## [1] 35.28688
```

Scatter plotting and calculating, on average, how far the predictions are from the median true home value,

```
plot(pred_2, test_2$medv)
abline(1:50, 1:50)
```




```
sqrt(mse_2_b)
```

```
## [1] 5.940276
```

Test MSE from the regression tree is 35.2868819. On average, the predictions are within 5.9402762 of the true value.

(c)

Q: Use the default settings and cross-validation in order to determine the optimal level of tree complexity. Would you prune the tree based on the result?

We could prune the tree as the number of splits which result in lowest `xerror` is given by a larger `cp` parameter. However, the differences are minimal, although a smaller tree is preferable to a larger tree. We decide to not prune the tree.

(d)

Q: Suppose you prune the tree to 5 nodes. Plot the new tree. What is the test set mean squared error? Compare with the result in part (b).

A: 5 nodes implies 4 splits. We hence need to get the corresponding `cp` value:

```
cp_val_2 <- tree_2_a$cptable[tree_2_a$cptable[,2] == 4, 1]
```

```
tree_2_d <- prune(tree_2_a, cp = cp_val_2)
pred_2_d <- predict(tree_2_d, newdata = test_2)
mse_2_d <- mean((pred_2_d - test_2$medv)^2)
mse_2_d
```

```
## [1] 35.90102
```

The test set mean squared error is now 35.9010228. This test error is higher than that in (b).

(e)

Q: Use random forests to analyze this data. Set the seed to 1 before running the method. What test set mean squared error do you obtain? How does this compare to the CART model? How many variables does the `randomForest` function try at each split?

A:

```
library(randomForest)
set.seed(1)
rf_2 <- randomForest(medv ~ ., data = train_2)
# rf_2
pred_2_e <- predict(rf_2, newdata = test_2)
mse_2_e <- mean((pred_2_e - test_2$medv)^2)
mse_2_e
```

```
## [1] 19.00006
```

The test MSE is 19.0000644. The result from the random forest, in terms of test error, is significantly smaller than CART. It tries 4 variables at each split by default.

(f)

Q: Use the `importance()` function to determine the two variables which are most important. Plot the importance measures using the `varImpPlot()`.

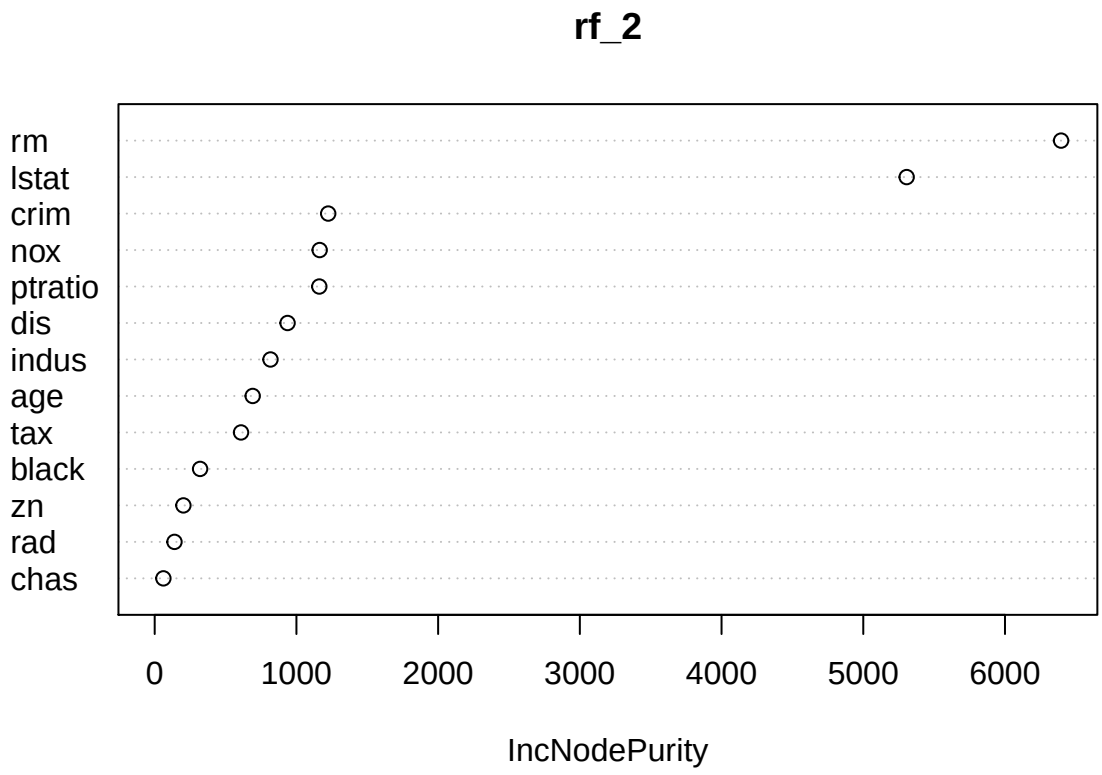
A: We can get the two most important variables, but we need to wrangle with the importance object a little:

```
import_2 <- as.vector(importance(rf_2))
names(import_2) <- rownames(importance((rf_2)))
names(sort(import_2, decreasing = TRUE)[1:2])
```

```
## [1] "rm"    "lstat"
```

Then plotting it,

```
varImpPlot(rf_2)
```



The two most important variables are `rm` and `lstat`.

(g)

Q: Describe the effect of the number of variables considered at each split controlled by the `mtry` argument in `randomForest()`, on the error obtained.

A: We can try out different values of `mtry` arguments in random forests to control this. Using all variables gives highly correlated trees, and as such, the prediction power of such a model would not be as good.

Question 3

(a)

Q: Read the data into the dataframe `supreme`. What is the fraction of cases in which the Supreme Court reversed the decision of the Lower Court?

A: Reading into the dataframe,

```
supreme <- read.csv("supremeexercise.csv")
```

The fraction of cases in which the Supreme Court reversed the decision can be directly calculated using a table,

```
table_3_a <- table(supreme$lctdir,supreme$result)
(table_3_a["conser", "0"] + table_3_a["liberal", "1"]) / sum(table_3_a)
```

```
## [1] 0.6153846
```

The fraction of cases in which the Supreme Court reversed the decision of the Lower Court is 0.6153846.

(b)

Q: Define a new variable `unCons` that takes a value of 1 if the decision made by the judges was an unanimous conservative decision and 0 otherwise. Write down the R command(s) that you used to define this variable. What is the total number of cases that had a unanimous conservative decision?

A:

```
supreme$unCons <- as.integer(rowSums(supreme[,5:13]) == 9)
# 5:13 are the judgments
table(supreme$unCons)
```

```
##
##   0   1
## 455 143
```

143 cases had a unanimous conservative decision.

(c)

Q: Define a new variable `unLib` that takes a value of 1 if the decision made by the judges was an unanimous liberal decision and 0 otherwise. What is the total number of cases that had an unanimous liberal decision?

A:

```
supreme$unLib <- as.integer(rowSums(supreme[,5:13]) == 0)
table(supreme$unLib)
```

```
##
##   0   1
## 474 124
```

124 cases had a unanimous liberal decision.

(d)

Q: You will now develop a two step CART model for this data. In the first step, you will build two classification trees to predict the unanimous conservative and liberal decisions respectively and in the second step, you will build nine judge-specific trees to predict the outcome for cases for which the predictions from the first step are ambiguous. Start by building a CART model to predict `unCons` using the six predictor variables `petit`, `respon`, `circuit`, `unconst`, `lctdir` and `issue`. Use the `rpart` package in R to build the model. Use the default parameter settings to build the CART model. Remember that you want to build a classification tree rather than a regression tree. Use all the observations to build the model. How many node splits are there in the resulting tree?

A:

```
library(rpart)
library(rpart.plot)
tree_3_d <- rpart((as.factor(unCons)~petit + respon +
                   circuit + unconst + lctdir + issue),
                  data = supreme)
# prp(tree_3_d)
unname(tail(tree_3_d$cptable[,2], 1))
```

```
## [1] 7
```

There are 7 node splits in the resulting tree.

(e)

Q: List all the variables that this tree splits on.

A:

```
spl_vars_3 <- as.character(unique(tree_3_d$frame[,1]))
spl_vars_3 <- spl_vars_3[!spl_vars_3 %in% "<leaf>"]
```

The tree in (d) splits on `circuit`, `issue`, `petit` and `respon`.

(f)

Q: What is the area under the curve for the receiver operating characteristic (ROC) curve for this model?

A:

```
library(ROCR)
pred_3_f <- predict(tree_3_d, newdata = supreme)
rocr_3_f <- prediction(pred_3_f[,2],supreme$unCons)
performance(rocr_3_f,"auc")@y.values
```

```
## [[1]]
## [1] 0.6519788
```

The AUC is given as 0.651978790440329.

(g)

Q: Similarly build a CART model to predict `unLib` using the six variables `petit`, `respon`, `circuit`, `unconst`, `lctdir` and `issue` as the predictor variables. Use the default parameter settings to build the CART model. Use all the observations to build the model. Which variable does the tree split on at the first level?

A:

```
tree_3_g <- rpart(as.factor(unLib)~petit+respon + circuit +
                unconst + lctdir + issue,
                data = supreme)
# prp(tree_3_g)
```

We can find what variable the tree splits on the first level directly:

```
tree_3_g$frame["1", 1]  # the string gets the row name
# referencing the row name directly is important when
# one wants to reference specific nodes directly e.g. "23"
```

```
## [1] respon
## Levels: <leaf> circuit issue petit respon
```

The tree splits on `respon` at the first level.

(h)

Q: Using the CART tree plot for the model in question (g), identify the leaf node with the fewest number of observations in it. What is the fraction of cases that has an unanimous liberal decision at this node?

A: To get the details of the node with the least number of observations, we can use the following:

```
tree_3_g$frame[which.min(tree_3_g$frame[,2]),]
```

```
##      var  n wt dev yval complexity ncompete nsurrogate   yval2.V1
## 6 <leaf> 11 11   3    1       0.01        0          0 1.00000000
##     yval2.V2   yval2.V3   yval2.V4   yval2.V5 yval2.nodeprob
## 6 8.00000000 3.00000000 0.72727273 0.27272727     0.01839465
```

What we are looking is located within the `frame` object, to access them directly we will be hardcoding in their positions with respect to the current version of `rpart` (`rpart_4.1-15`):

```
tree_3_g$frame[which.min(tree_3_g$frame[,2]),][[9]][3]/min(tree_3_g$frame[,2])
```

```
## [1] 0.2727273
```

The fraction of cases with unanimous liberal decisions at this node is $\frac{3}{11} = 0.2727273$.

(i)

Q: We will now combine the results from the two trees. What is the total number of cases where the two trees predict an unanimous outcome for the conservative and liberal judgement simultaneously, thus contradicting each other?

A: Predicting first, then getting the number of cases,

```
pred_3_i1 <- predict(tree_3_d, newdata = supreme, type = "class")
pred_3_i2 <- predict(tree_3_g, newdata = supreme, type = "class")
table(pred_3_i1, pred_3_i2) ["1", "1"]
```

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

The total number of cases where the two trees predict a unanimous outcome for the conservative and liberal judgment simultaneously is 2.

(j)

Q: What is the total number of cases where neither tree predicts an unanimous outcome?

A:

```
table(pred_3_i1, pred_3_i2) ["0", "0"]
```

```
## [1] 502
```

The total number of cases where neither tree predicts an unanimous outcome is 502.

(k)

Q: We now build the second part of our model which is nine judge-specific classification trees to provide predictions for the cases when either both trees predict an unanimous outcome or neither does (the harder cases). Build a CART model to predict each of the variables `rehndir` up to `brydir` using the six predictor variables `petit`, `respon`, `circuit`, `unconst`, `lctdir` and `issue`. Build you model using only those cases identified in questions (i) and (j). Use the majority of the judge predictions to make a prediction for each of these cases.

What is the accuracy of the model on these cases?

A:

```
judges_3 <- colnames(supreme)[5:13]
df_3_k <- subset(supreme, pred_3_i1 == pred_3_i2)
# conservative decisiosn
total_cons_3 <- integer(nrow(df_3_k))  # initialise as vector of 0s
for (judge in judges_3) {
  formula_i <- as.formula(paste0("as.factor(", judge, ")",
                                 "~ petit + respon + ",
                                 "circuit + unconst + ",
                                 "lctdir+issue"))
  tree_i <- rpart(formula_i, data = df_3_k)
  pred_i <- predict(tree_i, newdata = df_3_k,
                    type = "class")
  total_cons_3 <- total_cons_3 + as.numeric(levels(pred_i))[as.integer(pred_i)]
}
table_3_k <- table(total_cons_3 >= 5, df_3_k$result)
sum(diag(table_3_k))/nrow(df_3_k)
```

```
## [1] 0.7142857
```

The accuracy of the model on these cases is 0.7142857

(l)

Q: What is the overall accuracy of your two step CART model?

A: We get the accuracy of step 1 of the process first, using the tree from (d) as when the value of the prediction is 1 and the value as denoted in the original data is 1, the prediction is correct. (while for the tree in (g), a predicted value of 1 should correspond to a value of 0 in the original data.)

```
df_3_l <- subset(supreme, pred_3_i1 != pred_3_i2)
pred_3_l <- predict(tree_3_d, newdata = df_3_l, type = "class")
table_3_l <- table(pred_3_l, df_3_l$result)
sum(diag(table_3_l))
```

```
## [1] 76
```

A total of 76 cases were predicted correctly in the first step of the model.

We can then calculate the overall accuracy:

```
sum(diag(table_3_k + table_3_l)) / nrow(supreme)
```

```
## [1] 0.729097
```

The overall accuracy of the two step CART model is 0.729097.

(m)

Q: We consider the Moseley versus Victoria Secret Catalogue case in 2002, the details of which are as follows: The owner of the "Victoria's Secret" **business** brought a trademark dilution action against Victor Moseley in the Lower Court. They claimed that the "Victoria's Secret" trademark was diluted and tarnished by Moseley's adult specialty business named "Victor's Secret". The U.S. Lower Court for the **Sixth Circuit** ruled the judgment in a **conservative** direction by ruling in favor of Victoria's Secret. Moseley petitioned against this decision to the Supreme Court. Use your two step model to predict the outcome of this case which deals with **economic activities**. You can look at the tree plots to make your conclusion.

A:

```
# one way to verify this is to check visually with the trees
# prp(tree_3_d)
# prp(tree_3_g)

# alternatively we will code in the new data and predict with our trees
data_3_m <- data.frame(petit = "BUSINESS", respon = "BUSINESS",
                       circuit = "6th", unconst = 0,
                       lctdir = "conser", issue = "ECN")
predict(tree_3_d, newdata = data_3_m, type = "class")
```

```
## 1
## 0
## Levels: 0 1
```

```
predict(tree_3_g, newdata = data_3_m, type = "class")
```

```
## 1
## 1
## Levels: 0 1
```

As can be seen, both of the trees in the first step of the process agree in that there will be no unanimous conservative decision but there would be a unanimous liberal decision. Hence, the prediction is that there would be a liberal decision.

(n)

Q: We will now build a random forest model directly to predict the outcome result using the six predictor variables `petit`, `respon`, `circuit`, `unconst`, `lctdir` and `issue`. Use all the observations to build you model. Use the default settings to build the random forest model. What is the accuracy of the model?

A:

```
library(randomForest)
set.seed(1)
rf_3 <- randomForest(as.factor(result)~issue + circuit + lctdir +
                     unconst + petit + respon,
                     data = supreme)
pred_3_n <- predict(rf_3, newdata = supreme)
sum(diag(table(pred_3_n, supreme$result)))/nrow(supreme)
```

```
## [1] 0.8963211
```

The accuracy of the model is 0.8963211.

(o)

Q: The CART model and the random forest models have their respective advantages. Briefly provide one reason each as to why the CART model might be preferred to the random forest model and one reason why the random forest model might be preferred to the CART model.

A: CART is more interpretable than the random forest, but the latter has a greater accuracy.