Assignment Chapter 8

Vienne Jetten

2023-12-01

```
library(rpart)
library(magrittr)
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(ggplot2)
library(rpart.plot)
library(C50)
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

Preperation

First, we will get the data ready for building the CART and C5.0 trees. A dataframe is filled with the data:

```
churn <- read.csv("~/Documents/Uni/BA1/Period 2/Knowledge Discovery and Data Visualization/KDD 1 /Datas
```

Next, we examine the data's structure:

```
str(churn)
```

```
## 'data.frame':    3333 obs. of  21 variables:
##  $ State         : chr  "KS" "OH" "NJ" "OH" ...
##  $ Account.Length: int  128 107 137 84 75 118 121 147 117 141 ...
##  $ Area.Code     : int  415 415 415 408 415 510 510 415 408 415 ...
##  $ Phone         : chr  "382-4657" "371-7191" "358-1921" "375-9999" ...
##  $ Int.l.Plan    : chr  "no" "no" "no" "yes" ...
##  $ VMail.Plan    : chr  "yes" "yes" "no" "no" ...
##  $ VMail.Message : int  25 26 0 0 0 0 24 0 0 37 ...
##  $ Day.Mins      : num  265 162 243 299 167 ...
##  $ Day.Calls     : int  110 123 114 71 113 98 88 79 97 84 ...
##  $ Day.Charge    : num  45.1 27.5 41.4 50.9 28.3 ...
##  $ Eve.Mins      : num  197.4 195.5 121.2 61.9 148.3 ...
##  $ Eve.Calls     : int  99 103 110 88 122 101 108 94 80 111 ...
##  $ Eve.Charge    : num  16.78 16.62 10.3 5.26 12.61 ...
##  $ Night.Mins    : num  245 254 163 197 187 ...
##  $ Night.Calls   : int  91 103 104 89 121 118 118 96 90 97 ...
##  $ Night.Charge  : num  11.01 11.45 7.32 8.86 8.41 ...
##  $ Intl.Mins     : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
##  $ Intl.Calls    : int  3 3 5 7 3 6 7 6 4 5 ...
##  $ Intl.Charge   : num  2.7 3.7 3.29 1.78 2.73 1.7 2.03 1.92 2.35 3.02 ...
##  $ CustServ.Calls: int  1 1 0 2 3 0 3 0 1 0 ...
##  $ Churn.        : chr  "False." "False." "False." "False." ...
```

Numerical Variables: Account Length, Area Code, Voice Mail messages, Day minutes and Day calls, Day charge, Evening minutes, Calls and charges, Night minutes calls and charges, Customer Service calls, and, lastly, International minutes, Calls and charges are among the sixteen numerical variables that we find in the data set.

However, we are aware that Area Code is actually a factor variable with three levels, as shown below, based on our earlier work with the churn dataset:

```
Area.Code <- churn[c(3)]

churn$Area.Code <- as.factor(churn$Area.Code)
```

It will thus be handled as a categorical variable going forward.

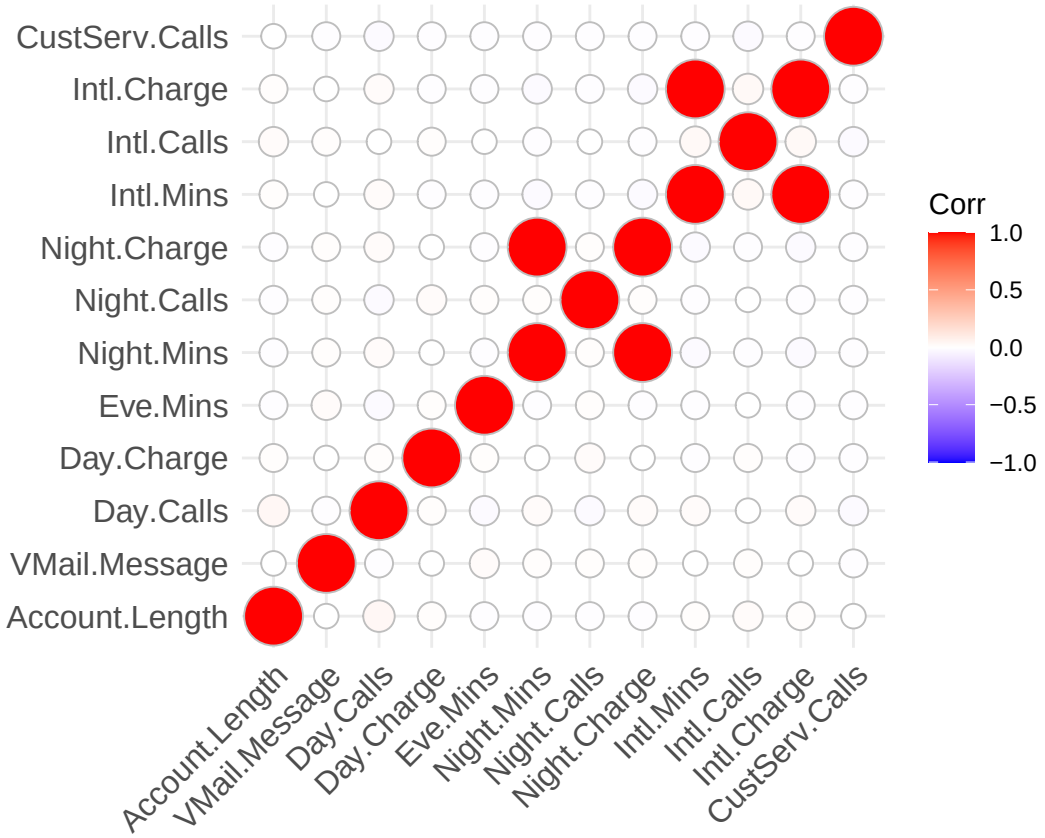
Initially, any correlations between any of the numerical variables will be looked for:

```
# Create a vector `con` containing column names of a data frame `churn`
con = c("Account.Length", "VMail.Message", "Day.Calls", "Day.Charge", "Eve.Mins",
        "Night.Mins", "Night.Calls", "Night.Charge", "Intl.Mins", "Intl.Calls",
        "Intl.Charge", "CustServ.Calls")

# Calculate the correlation matrix between columns in the `con` vector
corr = cor(churn[,con])

# Print the correlation matrix
corr
```

```
##                 Account.Length VMail.Message    Day.Calls    Day.Charge
```

```
## Account.Length    1.000000000 -0.0046278243  0.038469882  0.0062141347
## VMail.Message    -0.004627824  1.0000000000 -0.009548068  0.0007755235
## Day.Calls         0.038469882 -0.0095480677  1.000000000  0.0067529620
## Day.Charge        0.006214135  0.0007755235  0.006752962  1.0000000000
## Eve.Mins         -0.006757142  0.0175620343 -0.021451408  0.0070496072
## Night.Mins       -0.008955192  0.0076811359  0.022937845  0.0043238794
## Night.Calls      -0.013176275  0.0071230629 -0.019556965  0.0229724195
## Night.Charge     -0.008959535  0.0076632904  0.022926638  0.0043008608
## Intl.Mins         0.009513902  0.0028561959  0.021564794 -0.0101568616
## Intl.Calls        0.020661428  0.0139573387  0.004574268  0.0080315720
## Intl.Charge       0.009545675  0.0028836579  0.021666095 -0.0100942572
## CustServ.Calls   -0.003795939 -0.0132625831 -0.018941930 -0.0134269694
##                    Eve.Mins   Night.Mins   Night.Calls Night.Charge
## Account.Length -0.006757142 -0.008955192 -0.0131762751 -0.008959535
## VMail.Message   0.017562034  0.007681136  0.0071230629  0.007663290
## Day.Calls      -0.021451408  0.022937845 -0.0195569654  0.022926638
## Day.Charge      0.007049607  0.004323879  0.0229724195  0.004300861
## Eve.Mins        1.000000000 -0.012583678  0.0075856431 -0.012592806
## Night.Mins     -0.012583678  1.000000000  0.0112038563  0.999999215
## Night.Calls     0.007585643  0.011203856  1.0000000000  0.011187820
## Night.Charge   -0.012592806  0.999999215  0.0111878197  1.000000000
## Intl.Mins      -0.011034714 -0.015207297 -0.0136049964 -0.015213526
## Intl.Calls      0.002541292 -0.012353432  0.0003045795 -0.012329215
## Intl.Charge    -0.011066621 -0.015179849 -0.0136301696 -0.015186139
## CustServ.Calls -0.012984553 -0.009287613 -0.0128019273 -0.009276954
##                  Intl.Mins    Intl.Calls  Intl.Charge CustServ.Calls
## Account.Length  0.009513902  0.0206614284  0.009545675   -0.003795939
## VMail.Message   0.002856196  0.0139573387  0.002883658   -0.013262583
## Day.Calls       0.021564794  0.0045742682  0.021666095   -0.018941930
## Day.Charge     -0.010156862  0.0080315720 -0.010094257   -0.013426969
## Eve.Mins       -0.011034714  0.0025412917 -0.011066621   -0.012984553
## Night.Mins     -0.015207297 -0.0123534324 -0.015179849   -0.009287613
## Night.Calls    -0.013604996  0.0003045795 -0.013630170   -0.012801927
## Night.Charge   -0.015213526 -0.0123292150 -0.015186139   -0.009276954
## Intl.Mins       1.000000000  0.0323038841  0.999992742   -0.009639680
## Intl.Calls      0.032303884  1.0000000000  0.032372145   -0.017560599
## Intl.Charge     0.999992742  0.0323721453  1.000000000   -0.009674732
## CustServ.Calls -0.009639680 -0.0175605992 -0.009674732    1.000000000
```

```
library(ggcorrplot)

# Install the required packages to use the `ggcorrplot` function
#install.packages("ggcorplot")


# Create a correlation plot using the `ggcorrplot` function
ggcorrplot(corr, method = "circle")
```



It is evident to us that there are some strong relationships among our data. Also, it is evident that the numerical variables do not exhibit any additional significant associations.

Finally, we eliminate from the data set every variable related to the Charge type.

```
churn.2 <- subset(churn, select = -c(10,13,16,19) )
```

All of the remaining numerical variables can now be normalized. To achieve that, we'll apply the max/min approach. The code is displayed below:

```
normalize <- function(x){
  rng <- range(x,na.rm = TRUE)
  (x - rng[1]) / (rng[2] - rng[1])
}

churn.2 <- churn.2 %>% mutate(
  Account.Length.MM = normalize(Account.Length),
  VMail.Message.MM = normalize(VMail.Message),
  Day.Mins.MM = normalize(Day.Mins),
  Day.Calls.MM = normalize(Day.Calls),
  Eve.Mins.MM = normalize(Eve.Mins),
  Eve.Calls.MM = normalize(Eve.Calls),
  Night.Mins.MM = normalize(Night.Mins),
  Night.Calls.MM = normalize(Night.Calls),
  Intl.Mins.MM = normalize(Intl.Mins),
  Intl.Calls.MM = normalize(Intl.Calls),
```

```
  CustServ.Calls.MM = normalize(CustServ.Calls))
churn.3 <- subset(churn.2, select = -c(2, 7:16) )
```

Categorical Variables The data set's categorical variables have a number of problems. First, since it has 3333 levels, which is equal to the number of customers, the Phone variable—which is only a unique identifier for each customer—will be eliminated since it is a useless variable.

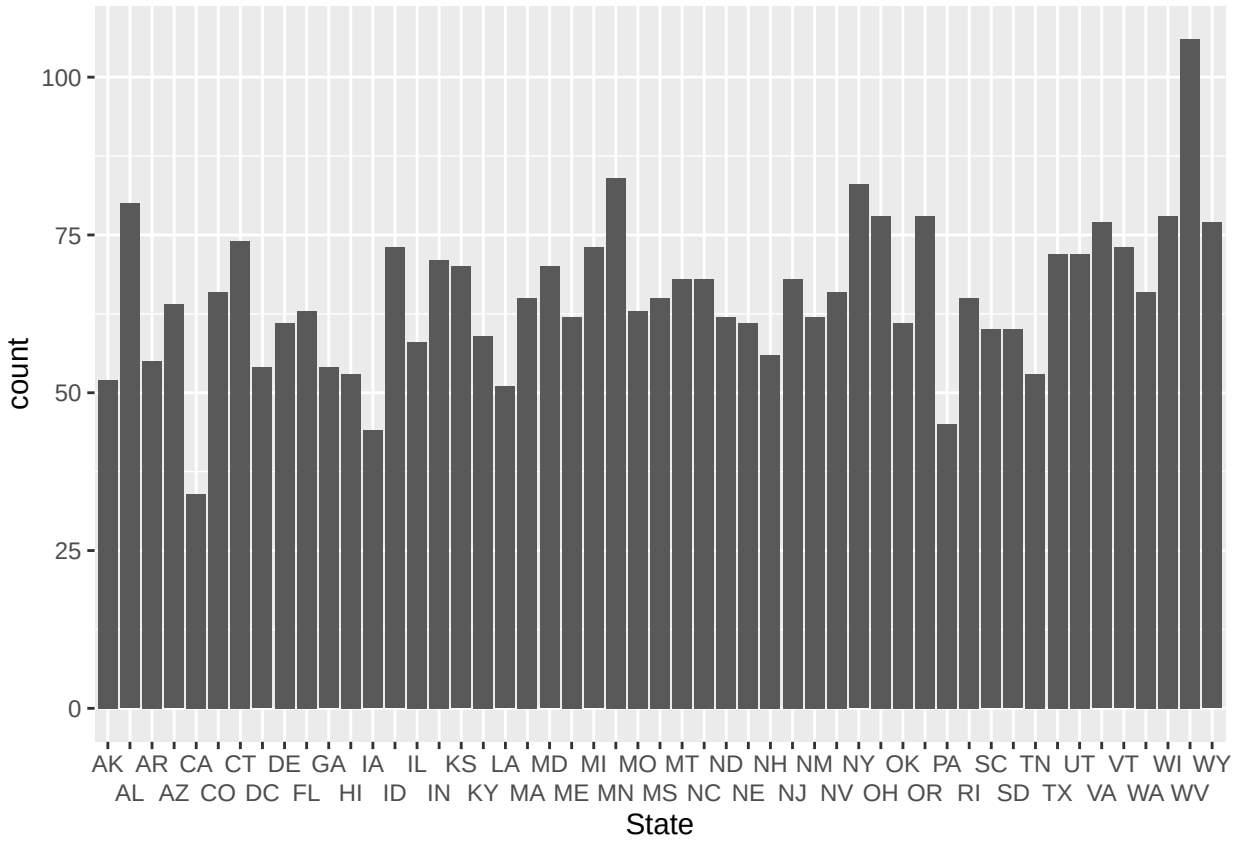
In a similar vein, the model will no longer include the State or Area Code variables. Only three Area Codes—all referring to California (CA)—are present in the datset despite the fact that there are 51 levels—more than there are States in the US. Furthermore, since West Virginia makes up the bulk of the states in the data set, it does not appear that the State or Area Code indicate the origin of a customer:

```
churn.3 <- churn.3 %>% mutate(
  Area.Code = as.factor(Area.Code))
churn.4 <- subset(churn.3, select = -c(1:3))
```

```
names(which.max(table(churn$State)))
```

```
## [1] "WV"
```

```
ggplot() + geom_bar(data=churn, aes(x=factor(State))) +
  scale_x_discrete(name ="State", guide = guide_axis(n.dodge=2)) +
  geom_bar(width = 0.6)
```



West Virginia (WV) is the majority class, this means that companies could focus on this state for more targeted retention strategies.

Since California is obviously the minority class and we do not have enough information on this variable, we are forced to eliminate both variables from the model due to the probability of anomalies within those fields.

The data is now prepared to develop a model. We may verify that by examining its composition:

```
str(churn.4)
```

```
## 'data.frame':    3333 obs. of  14 variables:
##  $ Int.l.Plan       : chr  "no" "no" "no" "yes" ...
##  $ VMail.Plan       : chr  "yes" "yes" "no" "no" ...
##  $ Churn.           : chr  "False." "False." "False." "False." ...
##  $ Account.Length.MM: num  0.525 0.438 0.562 0.343 0.306 ...
##  $ VMail.Message.MM : num  0.49 0.51 0 0 0 ...
##  $ Day.Mins.MM      : num  0.756 0.461 0.694 0.853 0.475 ...
##  $ Day.Calls.MM     : num  0.667 0.745 0.691 0.43 0.685 ...
##  $ Eve.Mins.MM      : num  0.543 0.538 0.333 0.17 0.408 ...
##  $ Eve.Calls.MM     : num  0.582 0.606 0.647 0.518 0.718 ...
##  $ Night.Mins.MM    : num  0.596 0.622 0.375 0.467 0.44 ...
##  $ Night.Calls.MM   : num  0.408 0.493 0.5 0.394 0.62 ...
##  $ Intl.Mins.MM     : num  0.5 0.685 0.61 0.33 0.505 0.315 0.375 0.355 0.435 0.56 ...
##  $ Intl.Calls.MM    : num  0.15 0.15 0.25 0.35 0.15 0.3 0.35 0.3 0.2 0.25 ...
##  $ CustServ.Calls.MM: num  0.111 0.111 0 0.222 0.333 ...
```

Exercise 11

Prior to creating the trees, the data was divided into two categories: training and holdout.

```
train_rows <- sample(nrow(churn.4), nrow(churn.4) * 0.75)
churn_train <- churn.4[train_rows, ]
churn_test <- churn.4[-train_rows, ]
```

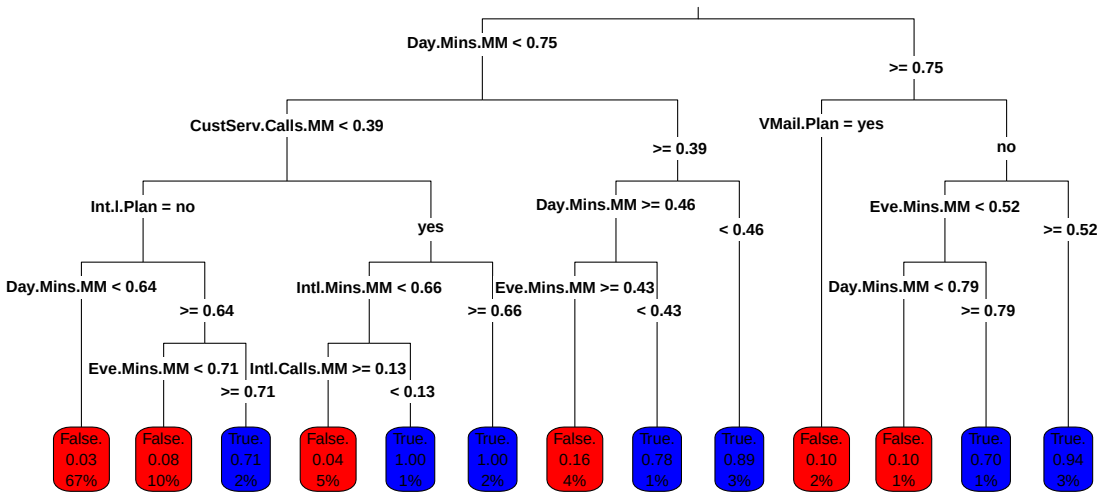
The following code generates a CART decision tree:

```
set.seed(11)

cartfit <- rpart(Churn.~. , data = churn_train, method = "class")

print(cartfit)
```

```
## n= 2499
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 2499 358 False. (0.85674270 0.14325730)
##    2) Day.Mins.MM< 0.7538483 2337 264 False. (0.88703466 0.11296534)
##      4) CustServ.Calls.MM< 0.3888889 2154 168 False. (0.92200557 0.07799443)
##        8) Int.l.Plan=no 1959  91 False. (0.95354773 0.04645227)
##         16) Day.Mins.MM< 0.6364025 1679  44 False. (0.97379392 0.02620608) *
##         17) Day.Mins.MM>=0.6364025 280  47 False. (0.83214286 0.16785714)
##           34) Eve.Mins.MM< 0.714325 242  20 False. (0.91735537 0.08264463) *
##           35) Eve.Mins.MM>=0.714325 38  11 True. (0.28947368 0.71052632) *
```

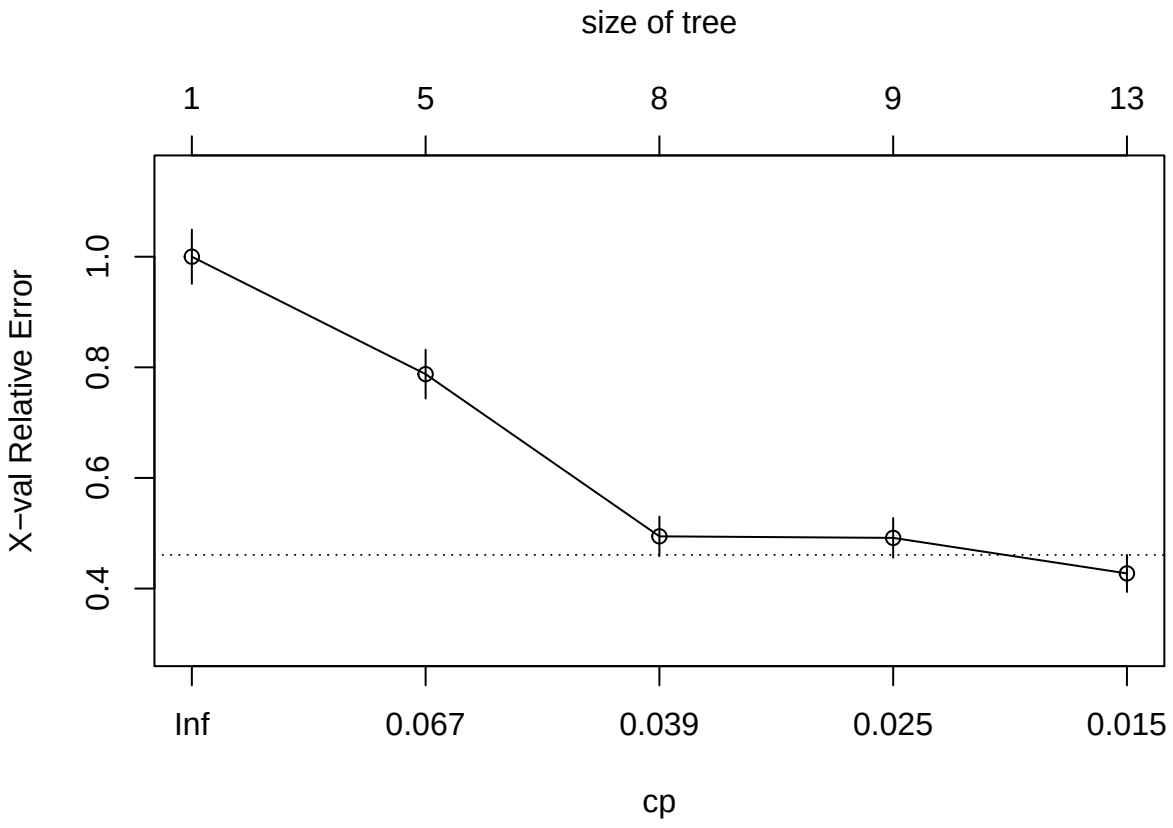
```
##         9) Int.l.Plan=yes 195  77 False. (0.60512821 0.39487179)
##          18) Intl.Mins.MM< 0.655 157  39 False. (0.75159236 0.24840764)
##            36) Intl.Calls.MM>=0.125 123   5 False. (0.95934959 0.04065041) *
##            37) Intl.Calls.MM< 0.125 34   0 True. (0.00000000 1.00000000) *
##          19) Intl.Mins.MM>=0.655 38   0 True. (0.00000000 1.00000000) *
##       5) CustServ.Calls.MM>=0.3888889 183  87 True. (0.47540984 0.52459016)
##        10) Day.Mins.MM>=0.4566705 107  28 False. (0.73831776 0.26168224)
##          20) Eve.Mins.MM>=0.4275502 89  14 False. (0.84269663 0.15730337) *
##          21) Eve.Mins.MM< 0.4275502 18   4 True. (0.22222222 0.77777778) *
##        11) Day.Mins.MM< 0.4566705 76   8 True. (0.10526316 0.89473684) *
##     3) Day.Mins.MM>=0.7538483 162  68 True. (0.41975309 0.58024691)
##       6) VMail.Plan=yes 42   4 False. (0.90476190 0.09523810) *
##       7) VMail.Plan=no 120  30 True. (0.25000000 0.75000000)
##        14) Eve.Mins.MM< 0.5169095 43  18 False. (0.58139535 0.41860465)
##          28) Day.Mins.MM< 0.7917617 20   2 False. (0.90000000 0.10000000) *
##          29) Day.Mins.MM>=0.7917617 23   7 True. (0.30434783 0.69565217) *
##        15) Eve.Mins.MM>=0.5169095 77   5 True. (0.06493506 0.93506494) *
```

```
rpart.plot(cartfit, type = 3, box.palette = c("red", "blue"), fallen.leaves = TRUE)
```



The figure below shows the tree's complexity (cp - complexity parameter), and the dotted line indicates the cp level at which pruning is recommended. This is so that overfitting may be prevented, but because it isn't required by the assignment, we won't perform it.

```
plotcp(cartfit)
```



We examine the CART tree's accuracy using the holdout set:

```
Churnpredict <- predict(cartfit, churn_test, type = "class")
mean(Churnpredict == churn_test$Churn.)
```

```
## [1] 0.9232614
```

This indicates the mean accuracy of the predictions. In this case, it's approximately 95.44%. This means that the model, when applied to the churn dataset, correctly predicted the outcome (churn or non-churn) for roughly 95.44% of the cases.

Exercise 12

Experienced trouble with creating a C4.5 tree because of Java issues. Instead we created a C5.0 tree. A C5.0 tree was generated using the code below:

```
set.seed(11)
predictors <- subset(churn_train, select = -c(3))
dependant <- churn_train$Churn.
dependant_factor <- as.factor(dependant)
c50fit <- C5.0(x = predictors, y = dependant_factor)
summary(c50fit)
```

```
##
## Call:
```

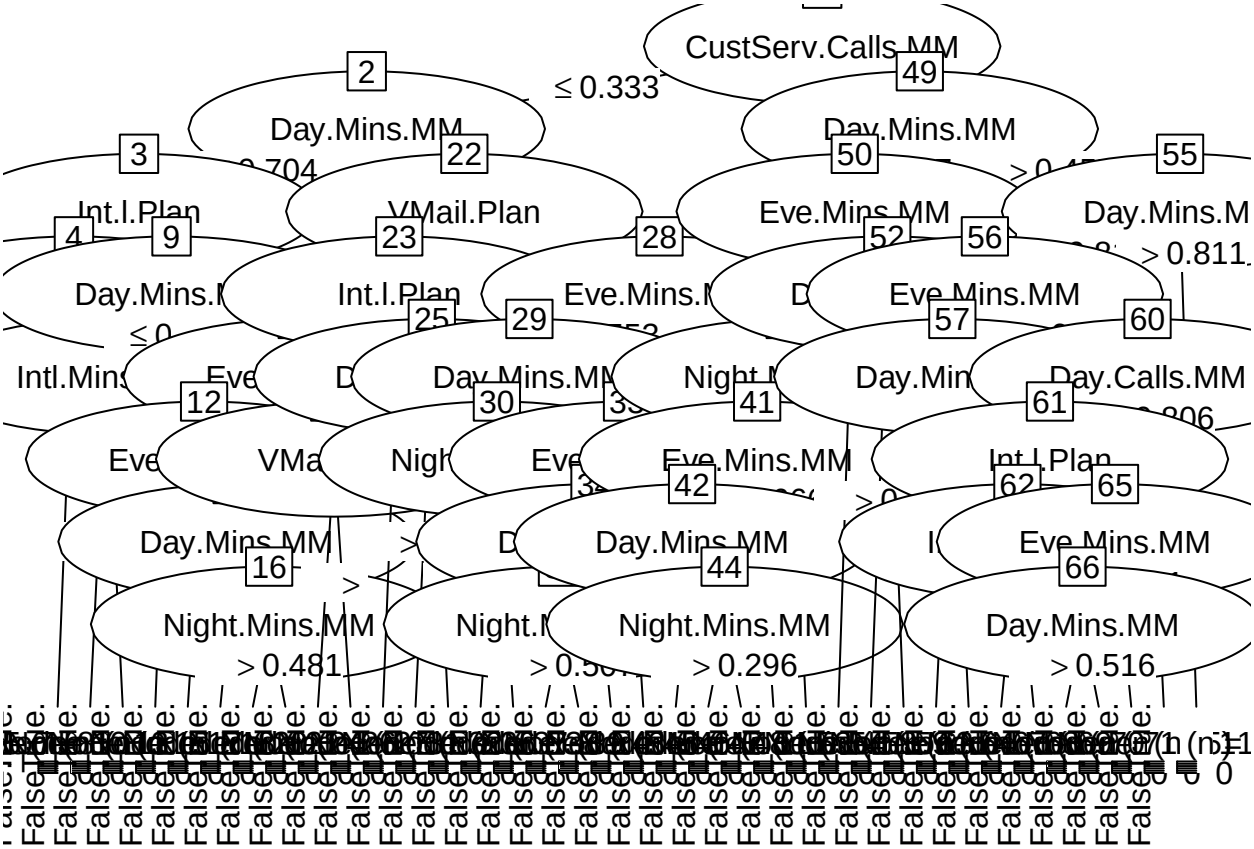
```
## C5.0.default(x = predictors, y = dependant_factor)
##
##
## C5.0 [Release 2.07 GPL Edition]      Mon Dec  4 13:56:13 2023
## -------------------------------
##
## Class specified by attribute 'outcome'
##
## Read 2499 cases (14 attributes) from undefined.data
##
## Decision tree:
##
## CustServ.Calls.MM <= 0.3333333:
## :...Day.Mins.MM <= 0.7041049:
## :   :...Int.l.Plan = yes:
## :   :   :...Intl.Calls.MM <= 0.1: True. (35)
## :   :   :   Intl.Calls.MM > 0.1:
## :   :   :   :...Intl.Mins.MM <= 0.655: False. (118/3)
## :   :   :       Intl.Mins.MM > 0.655: True. (32)
## :   :   Int.l.Plan = no:
## :   :   :...Day.Mins.MM <= 0.63626: False. (1679/44)
## :   :       Day.Mins.MM > 0.63626:
## :   :       :...Eve.Mins.MM > 0.7343965:
## :   :           :...VMail.Plan = no: True. (18/2)
## :   :           :   VMail.Plan = yes: False. (4)
## :   :           Eve.Mins.MM <= 0.7343965:
## :   :           :...Eve.Mins.MM <= 0.6648337: False. (155/5)
## :   :               Eve.Mins.MM > 0.6648337:
## :   :               :...Day.Mins.MM <= 0.6795895: False. (15)
## :   :                   Day.Mins.MM > 0.6795895:
## :   :                   :...Night.Mins.MM <= 0.4811727: False. (4)
## :   :                       Night.Mins.MM > 0.4811727: True. (6)
## :   Day.Mins.MM > 0.7041049:
## :   :...VMail.Plan = yes:
## :       :...Int.l.Plan = no: False. (57/1)
## :       :   Int.l.Plan = yes:
## :       :   :...Day.Mins.MM <= 0.7981756: True. (4)
## :       :       Day.Mins.MM > 0.7981756: False. (2)
## :       VMail.Plan = no:
## :       :...Eve.Mins.MM <= 0.5526533:
## :           :...Day.Mins.MM <= 0.7850627:
## :           :   :...Night.Mins.MM <= 0.7105971: False. (50/3)
## :           :   :   Night.Mins.MM > 0.7105971: True. (3)
## :           :   Day.Mins.MM > 0.7850627:
## :           :   :...Eve.Mins.MM > 0.4599945: True. (15)
## :           :       Eve.Mins.MM <= 0.4599945:
## :           :       :...Day.Mins.MM > 0.872862: True. (4)
## :           :           Day.Mins.MM <= 0.872862:
## :           :           :...Night.Mins.MM <= 0.5067241: False. (9)
## :           :               Night.Mins.MM > 0.5067241: True. (2)
## :           Eve.Mins.MM > 0.5526533:
## :           :...Night.Mins.MM > 0.4486283: True. (53)
## :               Night.Mins.MM <= 0.4486283:
## :               :...Eve.Mins.MM > 0.6664833: True. (17)
```

```
## :                   Eve.Mins.MM <= 0.6664833:
## :                   :...Day.Mins.MM <= 0.75: False. (9)
## :                       Day.Mins.MM > 0.75:
## :                       :...Night.Mins.MM <= 0.295589: False. (4)
## :                           Night.Mins.MM > 0.295589: True. (11)
## CustServ.Calls.MM > 0.3333333:
## :...Day.Mins.MM <= 0.4566705:
##     :...Eve.Mins.MM <= 0.6387132: True. (59/1)
##     :   Eve.Mins.MM > 0.6387132:
##     :   :...Day.Mins.MM <= 0.3429304: True. (8)
##     :       Day.Mins.MM > 0.3429304: False. (9/2)
##     Day.Mins.MM > 0.4566705:
##     :...Day.Mins.MM > 0.8107184: True. (6)
##         Day.Mins.MM <= 0.8107184:
##         :...Eve.Mins.MM <= 0.4275502:
##             :...Day.Mins.MM <= 0.6664766: True. (16/2)
##             :   Day.Mins.MM > 0.6664766: False. (2)
##             Eve.Mins.MM > 0.4275502:
##             :...Day.Calls.MM > 0.8060606: True. (5/1)
##                 Day.Calls.MM <= 0.8060606:
##                 :...Int.l.Plan = yes:
##                     :...Intl.Mins.MM <= 0.655: False. (7)
##                     :   Intl.Mins.MM > 0.655: True. (3)
##                     Int.l.Plan = no:
##                     :...Eve.Mins.MM > 0.5507286: False. (50/1)
##                         Eve.Mins.MM <= 0.5507286:
##                         :...Day.Mins.MM <= 0.5156785: True. (9/2)
##                             Day.Mins.MM > 0.5156785: False. (19/1)
##
##
## Evaluation on training data (2499 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##      36   68( 2.7%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##    2133     8    (a): class False.
##      60   298    (b): class True.
##
##
##  Attribute usage:
##
##  100.00% Day.Mins.MM
##  100.00% CustServ.Calls.MM
##   88.72% Int.l.Plan
##   22.65% Eve.Mins.MM
##   10.48% VMail.Plan
##    7.40% Intl.Calls.MM
##    6.72% Night.Mins.MM
```

```
##    6.40% Intl.Mins.MM
##    3.72% Day.Calls.MM
##
##
## Time: 0.0 secs
```

```
plot(c50fit)
```

```
## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf, breaks_split(split),
## : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf, breaks_split(split),
## : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf, breaks_split(split),
## : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf, breaks_split(split),
## : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf, breaks_split(split),
## : NAs introduced by coercion
```



There is a noticeable difference between the two just by seeing the plot of the C5.0 tree; this will be covered in more detail in 13.

We now evaluate the tree's accuracy using the holdout data:

```
Churnpredict.c50 <- predict(c50fit, churn_test, type = "class")
mean(Churnpredict.c50 == churn_test$Churn.)
```

```
## [1] 0.9364508
```

It is evident that the C5.0 tree is somewhat more accurate than the CART tree.

Exercise 13

When we compare the trees, we can observe that the C5.0 method produces a "bushier" tree—that is, a tree with a lot more splits and nodes—that is both broader and more complicated. This results from the notable variations in how each method builds a tree. While the C5.0 method is not limited to binary splits at each node and may also do non-binary splits when necessary, the CART tree display shows only binary splits at each node, giving rise to a "wider" tree. Due to their greater complexity, this generally results in a little higher accuracy, as seen above; nevertheless, this is a negative in and of itself because C5.0 trees are difficult to comprehend and grasp. The dividing criteria represent another significant distinction between the two algorithms. Information gain, or the amount of "harmful" noise we eliminate when transmitting a signal, is what drives C5.0. It is connected to how few bits we need to convey the information. A completely different split criterion is used by the CART method, which is predicated on producing child sets that are as "pure" and about equal in size.

CART: + Easier to visualize + Less complex - Frequently less accurate

C5.0: + Precision + Depth - Complexity - Hard to visualize

Exercise 14

The code below represents the CART tree as a set of rules:

```
rules <- rpart.rules(cartfit)
asRules(cartfit)
```

```
##
##  Rule number: 19 [Churn.=True. cover=38 (2%) prob=1.00]
##    Day.Mins.MM< 0.7538
##    CustServ.Calls.MM< 0.3889
##    Int.l.Plan=yes
##    Intl.Mins.MM>=0.655
##
##  Rule number: 37 [Churn.=True. cover=34 (1%) prob=1.00]
##    Day.Mins.MM< 0.7538
##    CustServ.Calls.MM< 0.3889
##    Int.l.Plan=yes
##    Intl.Mins.MM< 0.655
##    Intl.Calls.MM< 0.125
##
##  Rule number: 15 [Churn.=True. cover=77 (3%) prob=0.94]
##    Day.Mins.MM>=0.7538
##    VMail.Plan=no
##    Eve.Mins.MM>=0.5169
##
##  Rule number: 11 [Churn.=True. cover=76 (3%) prob=0.89]
##    Day.Mins.MM< 0.7538
##    CustServ.Calls.MM>=0.3889
##    Day.Mins.MM< 0.4567
##
##  Rule number: 21 [Churn.=True. cover=18 (1%) prob=0.78]
##    Day.Mins.MM< 0.7538
##    CustServ.Calls.MM>=0.3889
##    Day.Mins.MM>=0.4567
##    Eve.Mins.MM< 0.4276
##
##  Rule number: 35 [Churn.=True. cover=38 (2%) prob=0.71]
##    Day.Mins.MM< 0.7538
##    CustServ.Calls.MM< 0.3889
##    Int.l.Plan=no
##    Day.Mins.MM>=0.6364
##    Eve.Mins.MM>=0.7143
##
##  Rule number: 29 [Churn.=True. cover=23 (1%) prob=0.70]
##    Day.Mins.MM>=0.7538
##    VMail.Plan=no
##    Eve.Mins.MM< 0.5169
##    Day.Mins.MM>=0.7918
##
##  Rule number: 20 [Churn.=False. cover=89 (4%) prob=0.16]
```

```
##    Day.Mins.MM< 0.7538
##    CustServ.Calls.MM>=0.3889
##    Day.Mins.MM>=0.4567
##    Eve.Mins.MM>=0.4276
##
##  Rule number: 28 [Churn.=False. cover=20 (1%) prob=0.10]
##    Day.Mins.MM>=0.7538
##    VMail.Plan=no
##    Eve.Mins.MM< 0.5169
##    Day.Mins.MM< 0.7918
##
##  Rule number: 6 [Churn.=False. cover=42 (2%) prob=0.10]
##    Day.Mins.MM>=0.7538
##    VMail.Plan=yes
##
##  Rule number: 34 [Churn.=False. cover=242 (10%) prob=0.08]
##    Day.Mins.MM< 0.7538
##    CustServ.Calls.MM< 0.3889
##    Int.l.Plan=no
##    Day.Mins.MM>=0.6364
##    Eve.Mins.MM< 0.7143
##
##  Rule number: 36 [Churn.=False. cover=123 (5%) prob=0.04]
##    Day.Mins.MM< 0.7538
##    CustServ.Calls.MM< 0.3889
##    Int.l.Plan=yes
##    Intl.Mins.MM< 0.655
##    Intl.Calls.MM>=0.125
##
##  Rule number: 16 [Churn.=False. cover=1679 (67%) prob=0.03]
##    Day.Mins.MM< 0.7538
##    CustServ.Calls.MM< 0.3889
##    Int.l.Plan=no
##    Day.Mins.MM< 0.6364
```

Exercise 15

```
dependant_factor <- factor(dependant)
c50fit <- C5.0(x = predictors, y = dependant_factor)
rule_c50fit <- C5.0(x = predictors, y = dependant_factor, rules = TRUE)
summary(rule_c50fit)
```

```
##
## Call:
## C5.0.default(x = predictors, y = dependant_factor, rules = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Mon Dec  4 13:56:15 2023
## -------------------------------
##
## Class specified by attribute 'outcome'
##
```

```
## Read 2499 cases (14 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (1679/44, lift 1.1)
##  Int.l.Plan = no
##  Day.Mins.MM <= 0.63626
##  CustServ.Calls.MM <= 0.3333333
##  ->  class False.  [0.973]
##
## Rule 2: (118/3, lift 1.1)
##  Int.l.Plan = yes
##  Day.Mins.MM <= 0.7041049
##  Intl.Mins.MM <= 0.655
##  Intl.Calls.MM > 0.1
##  CustServ.Calls.MM <= 0.3333333
##  ->  class False.  [0.967]
##
## Rule 3: (153/8, lift 1.1)
##  Day.Mins.MM > 0.3429304
##  Day.Mins.MM <= 0.4566705
##  Eve.Mins.MM > 0.6387132
##  ->  class False.  [0.942]
##
## Rule 4: (1571/243, lift 1.0)
##  Day.Mins.MM > 0.4566705
##  ->  class False.  [0.845]
##
## Rule 5: (65, lift 6.9)
##  VMail.Plan = no
##  Day.Mins.MM > 0.75
##  Eve.Mins.MM > 0.5526533
##  Night.Mins.MM > 0.295589
##  ->  class True.  [0.985]
##
## Rule 6: (53, lift 6.9)
##  VMail.Plan = no
##  Day.Mins.MM > 0.7041049
##  Eve.Mins.MM > 0.5526533
##  Night.Mins.MM > 0.4486283
##  CustServ.Calls.MM <= 0.3333333
##  ->  class True.  [0.982]
##
## Rule 7: (46, lift 6.8)
##  Int.l.Plan = yes
##  Intl.Mins.MM > 0.655
##  ->  class True.  [0.979]
##
## Rule 8: (44, lift 6.8)
##  Int.l.Plan = yes
##  Intl.Calls.MM <= 0.1
##  ->  class True.  [0.978]
##
## Rule 9: (43, lift 6.8)
```

```
##  VMail.Plan = no
##  Day.Mins.MM > 0.7041049
##  Eve.Mins.MM > 0.6664833
##  ->  class True.  [0.978]
##
## Rule 10: (32, lift 6.8)
##  VMail.Plan = no
##  Day.Mins.MM > 0.7850627
##  Night.Mins.MM > 0.5067241
##  ->  class True.  [0.971]
##
## Rule 11: (59/1, lift 6.8)
##  Day.Mins.MM <= 0.4566705
##  Eve.Mins.MM <= 0.6387132
##  CustServ.Calls.MM > 0.3333333
##  ->  class True.  [0.967]
##
## Rule 12: (54/1, lift 6.7)
##  VMail.Plan = no
##  Day.Mins.MM > 0.7850627
##  Eve.Mins.MM > 0.4599945
##  ->  class True.  [0.964]
##
## Rule 13: (24, lift 6.7)
##  Day.Mins.MM <= 0.3429304
##  CustServ.Calls.MM > 0.3333333
##  ->  class True.  [0.962]
##
## Rule 14: (18, lift 6.6)
##  VMail.Plan = no
##  Day.Mins.MM > 0.872862
##  ->  class True.  [0.950]
##
## Rule 15: (35/2, lift 6.4)
##  VMail.Plan = no
##  Day.Mins.MM > 0.63626
##  Eve.Mins.MM > 0.7343965
##  ->  class True.  [0.919]
##
## Rule 16: (57/4, lift 6.4)
##  Day.Mins.MM <= 0.5156785
##  Eve.Mins.MM <= 0.5507286
##  CustServ.Calls.MM > 0.3333333
##  ->  class True.  [0.915]
##
## Rule 17: (28/2, lift 6.3)
##  Day.Mins.MM <= 0.6664766
##  Eve.Mins.MM <= 0.4275502
##  CustServ.Calls.MM > 0.3333333
##  ->  class True.  [0.900]
##
## Rule 18: (7, lift 6.2)
##  VMail.Plan = no
##  Day.Mins.MM > 0.7041049
```

```
##  Night.Mins.MM > 0.7105971
##  ->  class True.  [0.889]
##
## Rule 19: (7, lift 6.2)
##  Day.Mins.MM > 0.6795895
##  Day.Mins.MM <= 0.7041049
##  Eve.Mins.MM > 0.6648337
##  Eve.Mins.MM <= 0.7343965
##  Night.Mins.MM > 0.4811727
##  ->  class True.  [0.889]
##
## Default class: False.
##
##
## Evaluation on training data (2499 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##      19   66( 2.6%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##    2133     8    (a): class False.
##      58   300    (b): class True.
##
##
##  Attribute usage:
##
##   98.96% Day.Mins.MM
##   77.75% CustServ.Calls.MM
##   75.31% Int.l.Plan
##   14.77% Eve.Mins.MM
##    6.56% Intl.Mins.MM
##    6.48% Intl.Calls.MM
##    5.32% VMail.Plan
##    4.08% Night.Mins.MM
##
##
## Time: 0.0 secs
```

Exercise 16

Many of the above highlighted aspects become increasingly clearer when comparing the two trees written as rules. In contrast to the C5.0, which is more sophisticated, the CART tree generates fewer rules, which implies fewer potential "leaf" or end nodes.

Once again, the CART rules are simple to follow. However, the C5.0 algorithm is even simpler to read and comprehend when it comes to the rule form. One example of this is the attribute usage, which estimates the number of splits in which an attribute was used based on the percentage of all instances that fell within the resultant nodes. For example, if an attribute is used frequently along "paths," but only in the "end," indicating that the children sets have few instances overall, it would not be assigned a high attribute usage.

Additionally, it gives the chance of belonging to a particular class (after applying a Laplace adjustment) and the odds of belonging to the target class (i.e., the class distribution in the given leaf node). Finally, it offers a "Lift" metric (ratio of the rule's estimated accuracy to the relative frequency of the predicted class in the training set) that quantifies the improvement in accuracy attained by the rule in comparison to a random guess.

Apart from the "paths," the CART algorithm also yields some other useful data. The "cover" shows how many instances of the training data fit within the specified end node; this information is also displayed immediately to the right in percentages. The likelihood of target class membership in the specified end node is shown by the "prob"; a number as far from 0.5 as feasible is preferred. A value of 1 indicates that all are churning, while a value of 0 indicates none are.

CART: + information on "coverage" of the rule + data on the imformation of class membership at the conclusion of the rule

C5.0: + lift information + Laplace Corrected probabilities + Attribute usage information + relatively easier to read + detailed breakdown of class membership at each leaf (odds)