There are lots of tools that can help you predict an  
outcome, or classify, but CHAID is especially good at helping you  
explain to **any audience** how the model arrives at it’s prediction or  
classification. It’s also incredibly robust from a statistical  
perspective, making almost no assumptions about your data for  
distribution or normality. This post I’ll focus on marrying CHAID with  
the awesome caret package  
to make our predicting easier and hopefully more accurate. Although not  
strictly necessary you’re probably best served by reading the original  
post first.

We’ve been using a dataset that comes to us from the [IBM Watson  
Project](https://www.ibm.com/communities/analytics/watson-analytics-blog/hr-employee-attrition/)  
and comes packaged with the rsample library. It’s a very practical and  
understandable dataset. A great use case for a tree based algorithm.  
Imagine yourself in a fictional company faced with the task of trying to  
figure out which employees you are going to “lose” a.k.a. attrition or  
turnover. There’s a steep cost involved in keeping good employees, and  
training and on-boarding can be expensive. Being able to predict  
attrition even a little bit better would save you lots of money and make  
the company better, especially if you can understand exactly what you  
have to “watch out for” that might indicate the person is a high risk to  
leave.

**Setup and library loading**

If you’ve never used CHAID before you may also not have partykit.  
CHAID isn’t on CRAN but I have commented out the install command  
below. You’ll also get a variety of messages, none of which is relevant  
to this example so I’ve suppressed them.

# install.packages("partykit")

# install.packages("CHAID", repos="http://R-Forge.R-project.org")

require(rsample) # for dataset and splitting also loads broom and tidyr

require(dplyr)

require(CHAID)

require(purrr) # we'll use it to consolidate some data

require(caret)

require(kableExtra) # just to make the output nicer

**Predicting attrition in a fictional company**

I spent a great deal  
of time explaining the mechanics of loading the data. This time we’ll  
race right through. If you need an explanation of what’s going on please  
refer back. I’ve embedded some comments in the code to follow along and  
changing the data frame name to newattrit is not strictly necessary it  
just mimics the last post.

str(attrition) # included in rsample

## 'data.frame': 1470 obs. of 31 variables:

## $ Age : int 41 49 37 33 27 32 59 30 38 36 ...

## $ Attrition : Factor w/ 2 levels "No","Yes": 2 1 2 1 1 1 1 1 1 1 ...

## $ BusinessTravel : Factor w/ 3 levels "Non-Travel","Travel\_Frequently",..: 3 2 3 2 3 2 3 3 2 3 ...

## $ DailyRate : int 1102 279 1373 1392 591 1005 1324 1358 216 1299 ...

## $ Department : Factor w/ 3 levels "Human\_Resources",..: 3 2 2 2 2 2 2 2 2 2 ...

## $ DistanceFromHome : int 1 8 2 3 2 2 3 24 23 27 ...

## $ Education : Ord.factor w/ 5 levels "Below\_College"<..: 2 1 2 4 1 2 3 1 3 3 ...

## $ EducationField : Factor w/ 6 levels "Human\_Resources",..: 2 2 5 2 4 2 4 2 2 4 ...

## $ EnvironmentSatisfaction : Ord.factor w/ 4 levels "Low"<"Medium"<..: 2 3 4 4 1 4 3 4 4 3 ...

## $ Gender : Factor w/ 2 levels "Female","Male": 1 2 2 1 2 2 1 2 2 2 ...

## $ HourlyRate : int 94 61 92 56 40 79 81 67 44 94 ...

## $ JobInvolvement : Ord.factor w/ 4 levels "Low"<"Medium"<..: 3 2 2 3 3 3 4 3 2 3 ...

## $ JobLevel : int 2 2 1 1 1 1 1 1 3 2 ...

## $ JobRole : Factor w/ 9 levels "Healthcare\_Representative",..: 8 7 3 7 3 3 3 3 5 1 ...

## $ JobSatisfaction : Ord.factor w/ 4 levels "Low"<"Medium"<..: 4 2 3 3 2 4 1 3 3 3 ...

## $ MaritalStatus : Factor w/ 3 levels "Divorced","Married",..: 3 2 3 2 2 3 2 1 3 2 ...

## $ MonthlyIncome : int 5993 5130 2090 2909 3468 3068 2670 2693 9526 5237 ...

## $ MonthlyRate : int 19479 24907 2396 23159 16632 11864 9964 13335 8787 16577 ...

## $ NumCompaniesWorked : int 8 1 6 1 9 0 4 1 0 6 ...

## $ OverTime : Factor w/ 2 levels "No","Yes": 2 1 2 2 1 1 2 1 1 1 ...

## $ PercentSalaryHike : int 11 23 15 11 12 13 20 22 21 13 ...

## $ PerformanceRating : Ord.factor w/ 4 levels "Low"<"Good"<"Excellent"<..: 3 4 3 3 3 3 4 4 4 3 ...

## $ RelationshipSatisfaction: Ord.factor w/ 4 levels "Low"<"Medium"<..: 1 4 2 3 4 3 1 2 2 2 ...

## $ StockOptionLevel : int 0 1 0 0 1 0 3 1 0 2 ...

## $ TotalWorkingYears : int 8 10 7 8 6 8 12 1 10 17 ...

## $ TrainingTimesLastYear : int 0 3 3 3 3 2 3 2 2 3 ...

## $ WorkLifeBalance : Ord.factor w/ 4 levels "Bad"<"Good"<"Better"<..: 1 3 3 3 3 2 2 3 3 2 ...

## $ YearsAtCompany : int 6 10 0 8 2 7 1 1 9 7 ...

## $ YearsInCurrentRole : int 4 7 0 7 2 7 0 0 7 7 ...

## $ YearsSinceLastPromotion : int 0 1 0 3 2 3 0 0 1 7 ...

## $ YearsWithCurrManager : int 5 7 0 0 2 6 0 0 8 7 ...

# the easy to convert because they are integers with less than 10 levels

attrition <- attrition %>%

mutate\_if(function(col) length(unique(col)) <= 10 & is.integer(col), as.factor)

# More difficult to get 5 levels

attrition$YearsSinceLastPromotion <- cut(

attrition$YearsSinceLastPromotion,

breaks = c(-1, 0.9, 1.9, 2.9, 30),

labels = c("Less than 1", "1", "2", "More than 2")

)

# everything else just five more or less even levels

attrition <- attrition %>%

mutate\_if(is.numeric, funs(cut\_number(., n=5)))

dim(attrition)

## [1] 1470 31

str(attrition)

## 'data.frame': 1470 obs. of 31 variables:

## $ Age : Factor w/ 5 levels "[18,29]","(29,34]",..: 4 5 3 2 1 2 5 2 3 3 ...

## $ Attrition : Factor w/ 2 levels "No","Yes": 2 1 2 1 1 1 1 1 1 1 ...

## $ BusinessTravel : Factor w/ 3 levels "Non-Travel","Travel\_Frequently",..: 3 2 3 2 3 2 3 3 2 3 ...

## $ DailyRate : Factor w/ 5 levels "[102,392]","(392,656]",..: 4 1 5 5 2 4 5 5 1 5 ...

## $ Department : Factor w/ 3 levels "Human\_Resources",..: 3 2 2 2 2 2 2 2 2 2 ...

## $ DistanceFromHome : Factor w/ 5 levels "[1,2]","(2,5]",..: 1 3 1 2 1 1 2 5 5 5 ...

## $ Education : Ord.factor w/ 5 levels "Below\_College"<..: 2 1 2 4 1 2 3 1 3 3 ...

## $ EducationField : Factor w/ 6 levels "Human\_Resources",..: 2 2 5 2 4 2 4 2 2 4 ...

## $ EnvironmentSatisfaction : Ord.factor w/ 4 levels "Low"<"Medium"<..: 2 3 4 4 1 4 3 4 4 3 ...

## $ Gender : Factor w/ 2 levels "Female","Male": 1 2 2 1 2 2 1 2 2 2 ...

## $ HourlyRate : Factor w/ 5 levels "[30,45]","(45,59]",..: 5 3 5 2 1 4 4 3 1 5 ...

## $ JobInvolvement : Ord.factor w/ 4 levels "Low"<"Medium"<..: 3 2 2 3 3 3 4 3 2 3 ...

## $ JobLevel : Factor w/ 5 levels "1","2","3","4",..: 2 2 1 1 1 1 1 1 3 2 ...

## $ JobRole : Factor w/ 9 levels "Healthcare\_Representative",..: 8 7 3 7 3 3 3 3 5 1 ...

## $ JobSatisfaction : Ord.factor w/ 4 levels "Low"<"Medium"<..: 4 2 3 3 2 4 1 3 3 3 ...

## $ MaritalStatus : Factor w/ 3 levels "Divorced","Married",..: 3 2 3 2 2 3 2 1 3 2 ...

## $ MonthlyIncome : Factor w/ 5 levels "[1.01e+03,2.7e+03]",..: 4 3 1 2 2 2 1 1 4 3 ...

## $ MonthlyRate : Factor w/ 5 levels "[2.09e+03,6.89e+03]",..: 4 5 1 5 3 3 2 3 2 3 ...

## $ NumCompaniesWorked : Factor w/ 10 levels "0","1","2","3",..: 9 2 7 2 10 1 5 2 1 7 ...

## $ OverTime : Factor w/ 2 levels "No","Yes": 2 1 2 2 1 1 2 1 1 1 ...

## $ PercentSalaryHike : Factor w/ 5 levels "[11,12]","(12,13]",..: 1 5 3 1 1 2 5 5 5 2 ...

## $ PerformanceRating : Ord.factor w/ 4 levels "Low"<"Good"<"Excellent"<..: 3 4 3 3 3 3 4 4 4 3 ...

## $ RelationshipSatisfaction: Ord.factor w/ 4 levels "Low"<"Medium"<..: 1 4 2 3 4 3 1 2 2 2 ...

## $ StockOptionLevel : Factor w/ 4 levels "0","1","2","3": 1 2 1 1 2 1 4 2 1 3 ...

## $ TotalWorkingYears : Factor w/ 5 levels "[0,5]","(5,8]",..: 2 3 2 2 2 2 4 1 3 4 ...

## $ TrainingTimesLastYear : Factor w/ 7 levels "0","1","2","3",..: 1 4 4 4 4 3 4 3 3 4 ...

## $ WorkLifeBalance : Ord.factor w/ 4 levels "Bad"<"Good"<"Better"<..: 1 3 3 3 3 2 2 3 3 2 ...

## $ YearsAtCompany : Factor w/ 5 levels "[0,2]","(2,5]",..: 3 4 1 4 1 3 1 1 4 3 ...

## $ YearsInCurrentRole : Factor w/ 5 levels "[0,1]","(1,2]",..: 3 4 1 4 2 4 1 1 4 4 ...

## $ YearsSinceLastPromotion : Factor w/ 4 levels "Less than 1",..: 1 2 1 4 3 4 1 1 2 4 ...

## $ YearsWithCurrManager : Factor w/ 5 levels "[0,1]","(1,2]",..: 4 4 1 1 2 4 1 1 5 4 ...

newattrit <- attrition %>%

select\_if(is.factor)

dim(newattrit)

## [1] 1470 31

Okay we have data on 1,470 employees. We have 30 potential predictor  
(features) or independent variables and the all important attrition  
variable which gives us a yes or no answer to the question of whether or  
not the employee left. We’re to build the most accurate predictive model  
we can that is also simple (parsimonious) and explainable. The  
predictors we have seem to be the sorts of data we might have on hand in  
our HR files and thank goodness are labelled in a way that makes them  
pretty self explanatory.

Last post we explored the control options and built predictive models  
like the one below.

CHAID Details

You can get a very brief summary of CHAID from wikipedia and mentions of it scattered about in places like Analytics Vidhya or Data Flair. If you prefer a more scholarly bent the original article can be found in places like JSTOR. As the name implies it is fundamentally based on the venerable Chi-square test – and while not the most powerful (in terms of detecting the smallest possible differences) or the fastest, it really is easy to manage and more importantly to tell the story after using it.

Compared to some other techniques it’s also quite simple to use, as I hope you’ll agree, by the end of these posts. It’s a very practical and understandable dataset. A great use case for a tree based algorithm. Imagine yourself in a fictional company faced with the task of trying to figure out which employees you are going to “lose” a.k.a. attrition or turnover. There’s a steep cost involved in keeping good employees and training and on-boarding can be expensive. Being able to predict attrition even a little bit better would save you lots of money and make the company better, especially if you can understand exactly what you have to “watch out” for that might indicate the person is a high risk to leave.

**Setup and library loading**

If you’ve never used CHAID before you may also not have partykit. CHAID isn’t on CRAN but I have commented out the install command below. You’ll also get a variety of messages, none of which is relevant to this example so I’ve suppressed them.

*# install.packages("partykit")*

*# install.packages("CHAID", repos="http://R-Forge.R-project.org")*

require(rsample) *# for dataset and splitting also loads broom and tidyr*

require(dplyr)

require(ggplot2)

theme\_set(theme\_bw()) *# set theme*

require(CHAID)

require(purrr)

require(caret)

**Predicting attrition in a fictional company**

Let’s load up the attrition dataset and take a look at the variables we have.

*# data(attrition)*

str(attrition)

## 'data.frame': 1470 obs. of 31 variables:

## $ Age : int 41 49 37 33 27 32 59 30 38 36 ...

## $ Attrition : Factor w/ 2 levels "No","Yes": 2 1 2 1 1 1 1 1 1 1 ...

## $ BusinessTravel : Factor w/ 3 levels "Non-Travel","Travel\_Frequently",..: 3 2 3 2 3 2 3 3 2 3 ...

## $ DailyRate : int 1102 279 1373 1392 591 1005 1324 1358 216 1299 ...

## $ Department : Factor w/ 3 levels "Human\_Resources",..: 3 2 2 2 2 2 2 2 2 2 ...

## $ DistanceFromHome : int 1 8 2 3 2 2 3 24 23 27 ...

## $ Education : Ord.factor w/ 5 levels "Below\_College"<..: 2 1 2 4 1 2 3 1 3 3 ...

## $ EducationField : Factor w/ 6 levels "Human\_Resources",..: 2 2 5 2 4 2 4 2 2 4 ...

## $ EnvironmentSatisfaction : Ord.factor w/ 4 levels "Low"<"Medium"<..: 2 3 4 4 1 4 3 4 4 3 ...

## $ Gender : Factor w/ 2 levels "Female","Male": 1 2 2 1 2 2 1 2 2 2 ...

## $ HourlyRate : int 94 61 92 56 40 79 81 67 44 94 ...

## $ JobInvolvement : Ord.factor w/ 4 levels "Low"<"Medium"<..: 3 2 2 3 3 3 4 3 2 3 ...

## $ JobLevel : int 2 2 1 1 1 1 1 1 3 2 ...

## $ JobRole : Factor w/ 9 levels "Healthcare\_Representative",..: 8 7 3 7 3 3 3 3 5 1 ...

## $ JobSatisfaction : Ord.factor w/ 4 levels "Low"<"Medium"<..: 4 2 3 3 2 4 1 3 3 3 ...

## $ MaritalStatus : Factor w/ 3 levels "Divorced","Married",..: 3 2 3 2 2 3 2 1 3 2 ...

## $ MonthlyIncome : int 5993 5130 2090 2909 3468 3068 2670 2693 9526 5237 ...

## $ MonthlyRate : int 19479 24907 2396 23159 16632 11864 9964 13335 8787 16577 ...

## $ NumCompaniesWorked : int 8 1 6 1 9 0 4 1 0 6 ...

## $ OverTime : Factor w/ 2 levels "No","Yes": 2 1 2 2 1 1 2 1 1 1 ...

## $ PercentSalaryHike : int 11 23 15 11 12 13 20 22 21 13 ...

## $ PerformanceRating : Ord.factor w/ 4 levels "Low"<"Good"<"Excellent"<..: 3 4 3 3 3 3 4 4 4 3 ...

## $ RelationshipSatisfaction: Ord.factor w/ 4 levels "Low"<"Medium"<..: 1 4 2 3 4 3 1 2 2 2 ...

## $ StockOptionLevel : int 0 1 0 0 1 0 3 1 0 2 ...

## $ TotalWorkingYears : int 8 10 7 8 6 8 12 1 10 17 ...

## $ TrainingTimesLastYear : int 0 3 3 3 3 2 3 2 2 3 ...

## $ WorkLifeBalance : Ord.factor w/ 4 levels "Bad"<"Good"<"Better"<..: 1 3 3 3 3 2 2 3 3 2 ...

## $ YearsAtCompany : int 6 10 0 8 2 7 1 1 9 7 ...

## $ YearsInCurrentRole : int 4 7 0 7 2 7 0 0 7 7 ...

## $ YearsSinceLastPromotion : int 0 1 0 3 2 3 0 0 1 7 ...

## $ YearsWithCurrManager : int 5 7 0 0 2 6 0 0 8 7 ...

Okay we have data on 1,470 employees. We have 30 potential predictor or independent variables and the all important attrition variable which gives us a yes or no answer to the question of whether or not the employee left. We’re to build the most accurate predictive model we can that is also simple (parsimonious) and explainable. The predictors we have seem to be the sorts of data we might have on hand in our HR files and thank goodness are labelled in a way that makes them pretty self explanatory.

The CHAID library in R requires that any variables that we enter as predictors be either nominal or ordinal variables (see ?CHAID::chaid), which in R speak means we have to get them in as either factor or ordered factor. The str command shows we have a bunch of variables which are of type integer. As it turns out moving from integer to factor is simple in terms of code but has to be thoughtful for substantive reasons. So let’s see how things breakdown.

attrition **%>%**

select\_if(is.factor) **%>%**

ncol

## [1] 15

attrition **%>%**

select\_if(is.numeric) **%>%**

ncol

## [1] 16

Hmmmm, 15 factors and 16 integers. Let’s explore further. Of the variables that are integers how many of them have a small number of values (a.k.a. levels) and can therefore be simply and easily converted to true factors. We’ll use a dplyr pipe to see how many have 5 or fewer levels and 10 or fewer levels.

attrition **%>%**

select\_if(**function**(col)

**length**(unique(col)) **<=** 5 **&** **is.integer**(col)) **%>%**

head

## JobLevel StockOptionLevel

## 1 2 0

## 2 2 1

## 4 1 0

## 5 1 0

## 7 1 1

## 8 1 0

attrition **%>%**

select\_if(**function**(col)

**length**(unique(col)) **<=** 10 **&** **is.integer**(col)) **%>%**

head

## JobLevel NumCompaniesWorked StockOptionLevel TrainingTimesLastYear

## 1 2 8 0 0

## 2 2 1 1 3

## 4 1 6 0 3

## 5 1 1 0 3

## 7 1 9 1 3

## 8 1 0 0 2

2 and 4 respectively. We can be pretty confident that converting these from integer to factor won’t lose much information. Simple to run a mutate operation across the 4 we have identified. Probably more elegant though to make it a mutate\_if. That way in the future we decide we like 4 or 7 or 122 as our criteria for the change we only have to change one number. The “if” variation is also less to type and less likely to make a manual mistake.

attrition **%>%**

mutate(

JobLevel **=** factor(JobLevel),

NumCompaniesWorked **=** factor(NumCompaniesWorked),

StockOptionLevel **=** factor(StockOptionLevel),

TrainingTimesLastYear **=** factor(TrainingTimesLastYear)

) **%>%**

str

## 'data.frame': 1470 obs. of 31 variables:

## $ Age : int 41 49 37 33 27 32 59 30 38 36 ...

## $ Attrition : Factor w/ 2 levels "No","Yes": 2 1 2 1 1 1 1 1 1 1 ...

## $ BusinessTravel : Factor w/ 3 levels "Non-Travel","Travel\_Frequently",..: 3 2 3 2 3 2 3 3 2 3 ...

## $ DailyRate : int 1102 279 1373 1392 591 1005 1324 1358 216 1299 ...

## $ Department : Factor w/ 3 levels "Human\_Resources",..: 3 2 2 2 2 2 2 2 2 2 ...

## $ DistanceFromHome : int 1 8 2 3 2 2 3 24 23 27 ...

## $ Education : Ord.factor w/ 5 levels "Below\_College"<..: 2 1 2 4 1 2 3 1 3 3 ...

## $ EducationField : Factor w/ 6 levels "Human\_Resources",..: 2 2 5 2 4 2 4 2 2 4 ...

## $ EnvironmentSatisfaction : Ord.factor w/ 4 levels "Low"<"Medium"<..: 2 3 4 4 1 4 3 4 4 3 ...

## $ Gender : Factor w/ 2 levels "Female","Male": 1 2 2 1 2 2 1 2 2 2 ...

## $ HourlyRate : int 94 61 92 56 40 79 81 67 44 94 ...

## $ JobInvolvement : Ord.factor w/ 4 levels "Low"<"Medium"<..: 3 2 2 3 3 3 4 3 2 3 ...

## $ JobLevel : Factor w/ 5 levels "1","2","3","4",..: 2 2 1 1 1 1 1 1 3 2 ...

## $ JobRole : Factor w/ 9 levels "Healthcare\_Representative",..: 8 7 3 7 3 3 3 3 5 1 ...

## $ JobSatisfaction : Ord.factor w/ 4 levels "Low"<"Medium"<..: 4 2 3 3 2 4 1 3 3 3 ...

## $ MaritalStatus : Factor w/ 3 levels "Divorced","Married",..: 3 2 3 2 2 3 2 1 3 2 ...

## $ MonthlyIncome : int 5993 5130 2090 2909 3468 3068 2670 2693 9526 5237 ...

## $ MonthlyRate : int 19479 24907 2396 23159 16632 11864 9964 13335 8787 16577 ...

## $ NumCompaniesWorked : Factor w/ 10 levels "0","1","2","3",..: 9 2 7 2 10 1 5 2 1 7 ...

## $ OverTime : Factor w/ 2 levels "No","Yes": 2 1 2 2 1 1 2 1 1 1 ...

## $ PercentSalaryHike : int 11 23 15 11 12 13 20 22 21 13 ...

## $ PerformanceRating : Ord.factor w/ 4 levels "Low"<"Good"<"Excellent"<..: 3 4 3 3 3 3 4 4 4 3 ...

## $ RelationshipSatisfaction: Ord.factor w/ 4 levels "Low"<"Medium"<..: 1 4 2 3 4 3 1 2 2 2 ...

## $ StockOptionLevel : Factor w/ 4 levels "0","1","2","3": 1 2 1 1 2 1 4 2 1 3 ...

## $ TotalWorkingYears : int 8 10 7 8 6 8 12 1 10 17 ...

## $ TrainingTimesLastYear : Factor w/ 7 levels "0","1","2","3",..: 1 4 4 4 4 3 4 3 3 4 ...

## $ WorkLifeBalance : Ord.factor w/ 4 levels "Bad"<"Good"<"Better"<..: 1 3 3 3 3 2 2 3 3 2 ...

## $ YearsAtCompany : int 6 10 0 8 2 7 1 1 9 7 ...

## $ YearsInCurrentRole : int 4 7 0 7 2 7 0 0 7 7 ...

## $ YearsSinceLastPromotion : int 0 1 0 3 2 3 0 0 1 7 ...

## $ YearsWithCurrManager : int 5 7 0 0 2 6 0 0 8 7 ...

attrition **<-** attrition **%>%**

mutate\_if(**function**(col) **length**(unique(col)) **<=** 10 **&** **is.integer**(col), as.factor)

summary(attrition)

## Age Attrition BusinessTravel DailyRate

## Min. :18.00 No :1233 Non-Travel : 150 Min. : 102.0

## 1st Qu.:30.00 Yes: 237 Travel\_Frequently: 277 1st Qu.: 465.0

## Median :36.00 Travel\_Rarely :1043 Median : 802.0

## Mean :36.92 Mean : 802.5

## 3rd Qu.:43.00 3rd Qu.:1157.0

## Max. :60.00 Max. :1499.0

##

## Department DistanceFromHome Education

## Human\_Resources : 63 Min. : 1.000 Below\_College:170

## Research\_Development:961 1st Qu.: 2.000 College :282

## Sales :446 Median : 7.000 Bachelor :572

## Mean : 9.193 Master :398

## 3rd Qu.:14.000 Doctor : 48

## Max. :29.000

##

## EducationField EnvironmentSatisfaction Gender

## Human\_Resources : 27 Low :284 Female:588

## Life\_Sciences :606 Medium :287 Male :882

## Marketing :159 High :453

## Medical :464 Very\_High:446

## Other : 82

## Technical\_Degree:132

##

## HourlyRate JobInvolvement JobLevel

## Min. : 30.00 Low : 83 1:543

## 1st Qu.: 48.00 Medium :375 2:534

## Median : 66.00 High :868 3:218

## Mean : 65.89 Very\_High:144 4:106

## 3rd Qu.: 83.75 5: 69

## Max. :100.00

##

## JobRole JobSatisfaction MaritalStatus

## Sales\_Executive :326 Low :289 Divorced:327

## Research\_Scientist :292 Medium :280 Married :673

## Laboratory\_Technician :259 High :442 Single :470

## Manufacturing\_Director :145 Very\_High:459

## Healthcare\_Representative:131

## Manager :102

## (Other) :215

## MonthlyIncome MonthlyRate NumCompaniesWorked OverTime

## Min. : 1009 Min. : 2094 1 :521 No :1054

## 1st Qu.: 2911 1st Qu.: 8047 0 :197 Yes: 416

## Median : 4919 Median :14236 3 :159

## Mean : 6503 Mean :14313 2 :146

## 3rd Qu.: 8379 3rd Qu.:20462 4 :139

## Max. :19999 Max. :26999 7 : 74

## (Other):234

## PercentSalaryHike PerformanceRating RelationshipSatisfaction

## Min. :11.00 Low : 0 Low :276

## 1st Qu.:12.00 Good : 0 Medium :303

## Median :14.00 Excellent :1244 High :459

## Mean :15.21 Outstanding: 226 Very\_High:432

## 3rd Qu.:18.00

## Max. :25.00

##

## StockOptionLevel TotalWorkingYears TrainingTimesLastYear WorkLifeBalance

## 0:631 Min. : 0.00 0: 54 Bad : 80

## 1:596 1st Qu.: 6.00 1: 71 Good :344

## 2:158 Median :10.00 2:547 Better:893

## 3: 85 Mean :11.28 3:491 Best :153

## 3rd Qu.:15.00 4:123

## Max. :40.00 5:119

## 6: 65

## YearsAtCompany YearsInCurrentRole YearsSinceLastPromotion

## Min. : 0.000 Min. : 0.000 Min. : 0.000

## 1st Qu.: 3.000 1st Qu.: 2.000 1st Qu.: 0.000

## Median : 5.000 Median : 3.000 Median : 1.000

## Mean : 7.008 Mean : 4.229 Mean : 2.188

## 3rd Qu.: 9.000 3rd Qu.: 7.000 3rd Qu.: 3.000

## Max. :40.000 Max. :18.000 Max. :15.000

##

## YearsWithCurrManager

## Min. : 0.000

## 1st Qu.: 2.000

## Median : 3.000

## Mean : 4.123

## 3rd Qu.: 7.000

## Max. :17.000

##

As you look at the results this is a good time to remind you that CHAID is “non parametric” which means that we don’t have to worry about how the distribution (normality) looks nor make any assumptions about the variance. We are assuming that the predictors are independent of one another, but that is true of every statistical test and this is a robust procedure. So for now, let’s simply ignore all the variables that are still integers. I promise we’ll come back and deal with them later. But for now I’m eager to actually use CHAID and do some predicting. We’re also going to defer and address the issue of “over-fitting” and how to most wisely use the data we have. We’re simply going to build a first model using all 1,470 cases, the 18 factors we have available to predict with and we are trying to predict attrition. We’ll create a new dataframe called newattrit (how original right?).

newattrit **<-** attrition **%>%**

select\_if(is.factor)

**dim**(newattrit)

## [1] 1470 19

The chaid command accepts two pieces of information in it’s simplest case, a formula like outcome ~ predictors and a dataframe. We’re going to make use of the ~ . shortcut on the right hand side and add attrition on the left and newattrit as our dataframe.

About 6 seconds later (at least on my Mac) we’ll have a solution that we can print and plot.

***I’m going to output all the plots in a smaller size for the benefit of you the readers. I’m doing that via RMarkdown and it won’t happen automatically for you if you download and use the code. I’ll initially be using, fig.height=10, fig.width=20, dpi=90, out.width=“900px”***

What does CHAID do? Straight from the help pages “Select the predictor that has the smallest adjusted p-value (i.e., most significant). If this adjusted p-value is less than or equal to a user-specified alpha-level alpha4, split the node using this predictor. Else, do not split and the node is considered as a terminal node.” So it will take our 18 predictors and test each one against our outcome variable – attrition. The one with the lowest p value (a proxy for is most predictive) will “anchor” our decision tree. It will then repeat this process of splitting until more splits fail to yield *significant* results. I’m way over-simplifying, of course, but you get the idea. The end result will be a series of terminal nodes (think of them as “prediction buckets” that have a group of employees who all meet the same criteria who we think will either attrit or not attrit). Let’s run it.

*# demonstrate a full model using chaid with defaults*

chaidattrit1 **<-** chaid(Attrition **~** ., data **=** newattrit)

print(chaidattrit1)

##

## Model formula:

## Attrition ~ BusinessTravel + Department + Education + EducationField +

## EnvironmentSatisfaction + Gender + JobInvolvement + JobLevel +

## JobRole + JobSatisfaction + MaritalStatus + NumCompaniesWorked +

## OverTime + PerformanceRating + RelationshipSatisfaction +

## StockOptionLevel + TrainingTimesLastYear + WorkLifeBalance

##

## Fitted party:

## [1] root

## | [2] OverTime in No

## | | [3] StockOptionLevel in 0

## | | | [4] JobSatisfaction in Low

## | | | | [5] RelationshipSatisfaction in Low, Medium, High: No (n = 56, err = 42.9%)

## | | | | [6] RelationshipSatisfaction in Very\_High: No (n = 28, err = 7.1%)

## | | | [7] JobSatisfaction in Medium, High

## | | | | [8] JobInvolvement in Low: Yes (n = 12, err = 41.7%)

## | | | | [9] JobInvolvement in Medium, High, Very\_High

## | | | | | [10] BusinessTravel in Non-Travel, Travel\_Rarely: No (n = 181, err = 9.9%)

## | | | | | [11] BusinessTravel in Travel\_Frequently

## | | | | | | [12] RelationshipSatisfaction in Low: Yes (n = 8, err = 25.0%)

## | | | | | | [13] RelationshipSatisfaction in Medium, High, Very\_High: No (n = 30, err = 16.7%)

## | | | [14] JobSatisfaction in Very\_High: No (n = 134, err = 7.5%)

## | | [15] StockOptionLevel in 1, 2, 3

## | | | [16] EnvironmentSatisfaction in Low: No (n = 127, err = 11.0%)

## | | | [17] EnvironmentSatisfaction in Medium, High, Very\_High

## | | | | [18] Department in Human\_Resources, Sales: No (n = 164, err = 8.5%)

## | | | | [19] Department in Research\_Development: No (n = 314, err = 3.2%)

## | [20] OverTime in Yes

## | | [21] JobLevel in 1

## | | | [22] StockOptionLevel in 0, 3

## | | | | [23] JobSatisfaction in Low, Medium, High: Yes (n = 61, err = 26.2%)

## | | | | [24] JobSatisfaction in Very\_High: No (n = 28, err = 46.4%)

## | | | [25] StockOptionLevel in 1, 2

## | | | | [26] BusinessTravel in Non-Travel, Travel\_Rarely: No (n = 50, err = 26.0%)

## | | | | [27] BusinessTravel in Travel\_Frequently: Yes (n = 17, err = 35.3%)

## | | [28] JobLevel in 2, 3, 4, 5

## | | | [29] MaritalStatus in Divorced, Married

## | | | | [30] EnvironmentSatisfaction in Low, Medium: No (n = 60, err = 20.0%)

## | | | | [31] EnvironmentSatisfaction in High, Very\_High

## | | | | | [32] TrainingTimesLastYear in 0, 6: No (n = 10, err = 40.0%)

## | | | | | [33] TrainingTimesLastYear in 1, 2, 3, 4, 5

## | | | | | | [34] EnvironmentSatisfaction in Low, Medium, High: No (n = 57, err = 0.0%)

## | | | | | | [35] EnvironmentSatisfaction in Very\_High: No (n = 61, err = 6.6%)

## | | | [36] MaritalStatus in Single

## | | | | [37] Department in Human\_Resources, Research\_Development: No (n = 37, err = 10.8%)

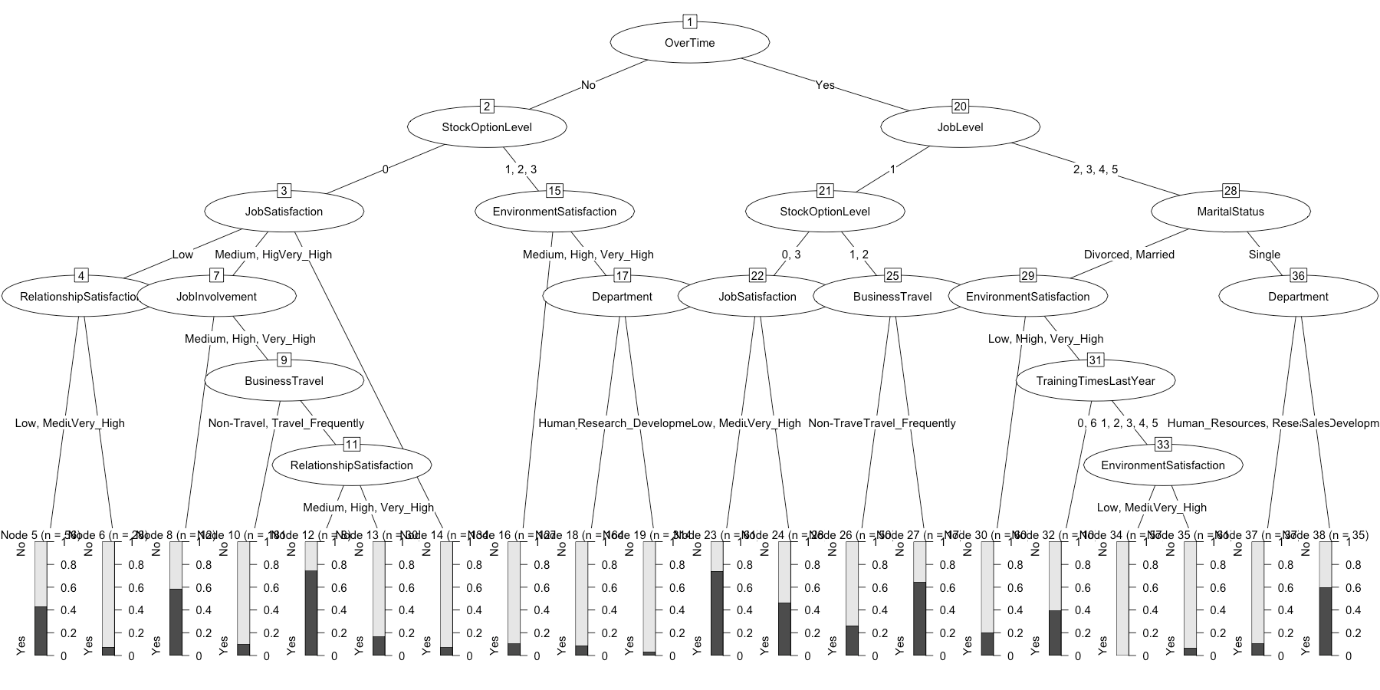
## | | | | [38] Department in Sales: Yes (n = 35, err = 40.0%)

##

## Number of inner nodes: 18

## Number of terminal nodes: 20

plot(chaidattrit1)



chisq.test(newattrit**$**Attrition, newattrit**$**OverTime)

##

## Pearson's Chi-squared test with Yates' continuity correction

##

## data: newattrit$Attrition and newattrit$OverTime

## X-squared = 87.564, df = 1, p-value < 2.2e-16

I happen to be a visual learner and prefer the plot to the print but they are obviously reporting the same information so use them as you see fit. As you can see the very first split it decides on is overtime yes or no. I’ve run the chi-square test so that you can see the p value is indeed very small (0.00000000000000022).

So the algorithm has decided that the most predictive way to divide our sample of employees is into 20 terminal nodes or buckets. Each one of the nodes represents a distinct set of predictors. Take a minute to look at node 19. Every person there shares the following characteristics.

* [2] OverTime in No
* [15] StockOptionLevel in 1, 2, 3
* [17] EnvironmentSatisfaction in Medium, High, Very\_High
* [19] Department in Research\_Development: No

There are n = 314 in this group, our prediction is that No they will not attrit and we were “wrong” err = 3.2%. That’s some useful information. To quote an old Star Wars movie “These are not the droids you’re looking for…”. In other words, this is not a group we should be overly worried about losing and we can say that with pretty high confidence.

For contrast let’s look at node #23:

* [20] OverTime in Yes
* [21] JobLevel in 1
* [22] StockOptionLevel in 0, 3
* [23] JobSatisfaction in Low, Medium, High:

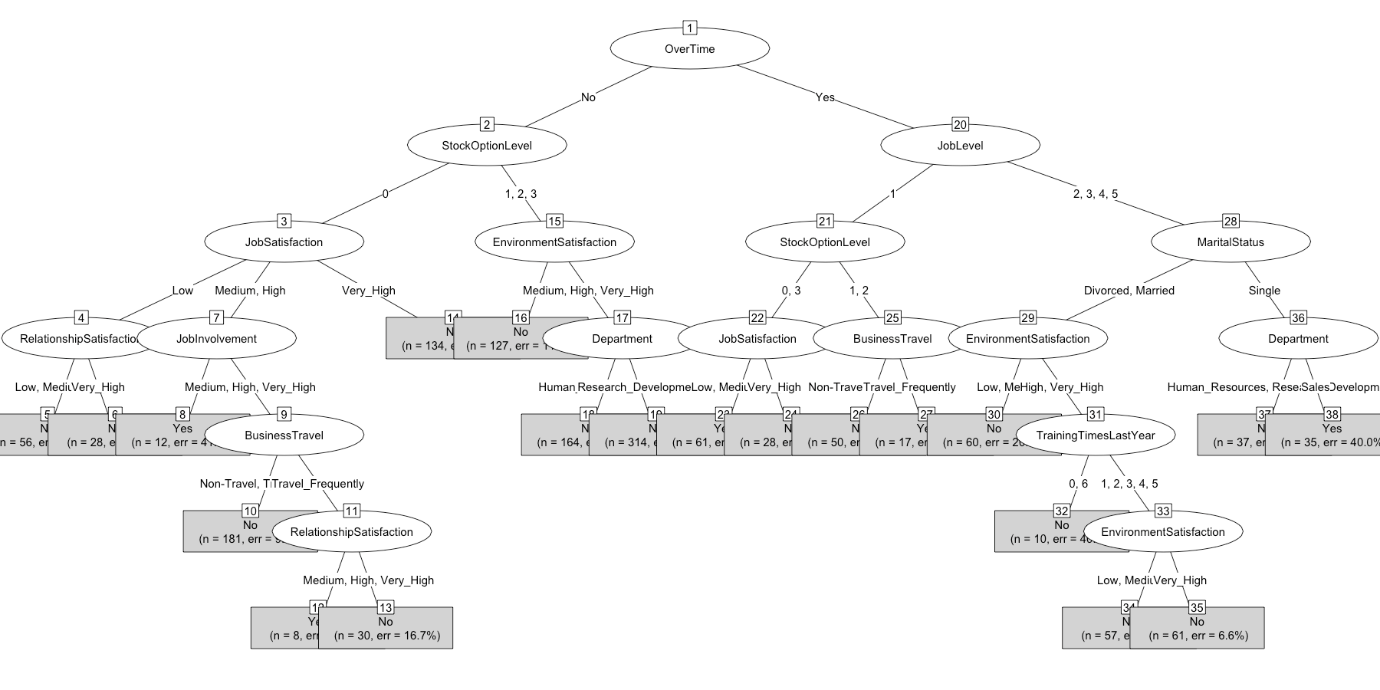
Where there are n = 61 staff, we predict they will leave Yes and we get it wrong err = 26.2% of the time. A little worrisome that we’re not as accurate but this is a group that bears watching or intervention if we want to retain them.

Some other things to note. Because the predictors are considered categorical we will get splits like we do for node 22, where 0 and 3 are on one side and 1, 2 is on the other. The number of people in any node can be quite variable. Finally, notice that a variable can occur at different levels of the model like StockOptionLevel does!

On the plot side of things there are a few key options you can adjust to make things easier to read. The next blocks of code show you how to adjust some key options such as adding a title, reducing the font size, using “simple” mode, and changing colors.

*# digress for plotting*

plot(chaidattrit1, type **=** "simple")



plot(

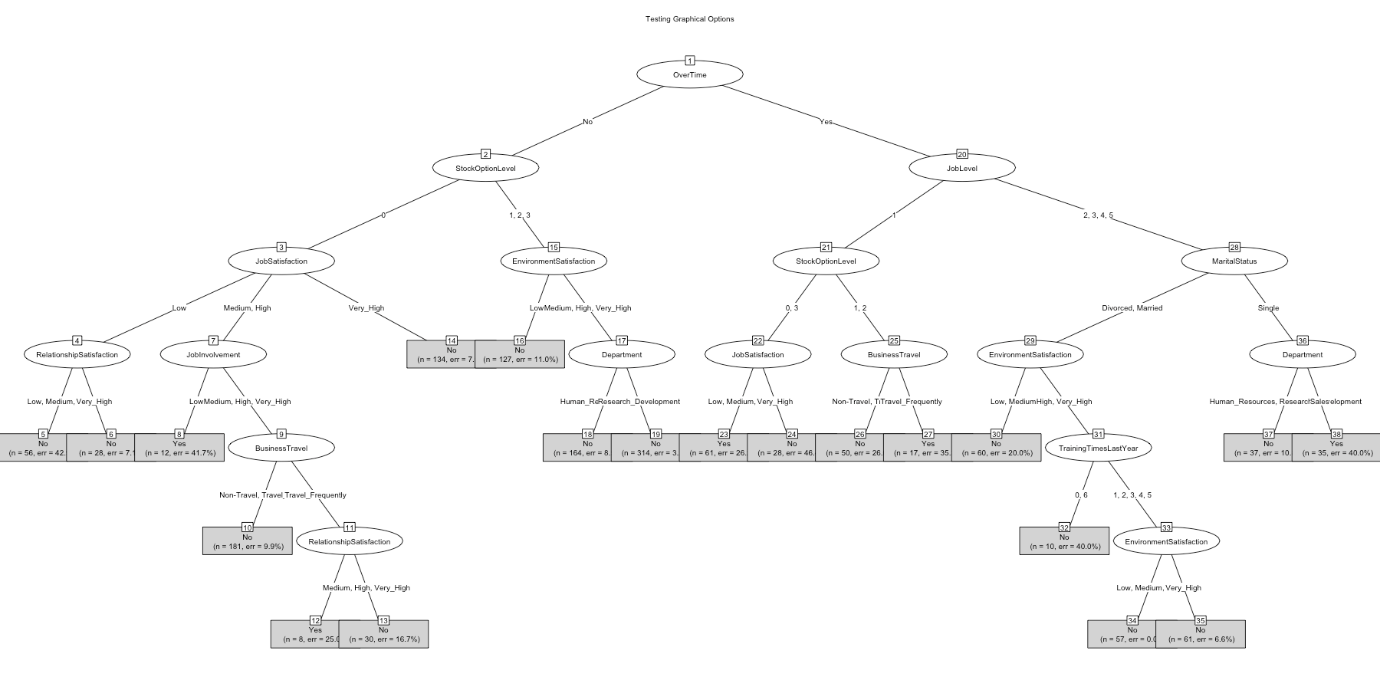
chaidattrit1,

main **=** "Testing Graphical Options",

gp **=** gpar(fontsize **=** 8),

type **=** "simple"

)



plot(

chaidattrit1,

main **=** "Testing More Graphical Options",

gp **=** gpar(

col **=** "blue",

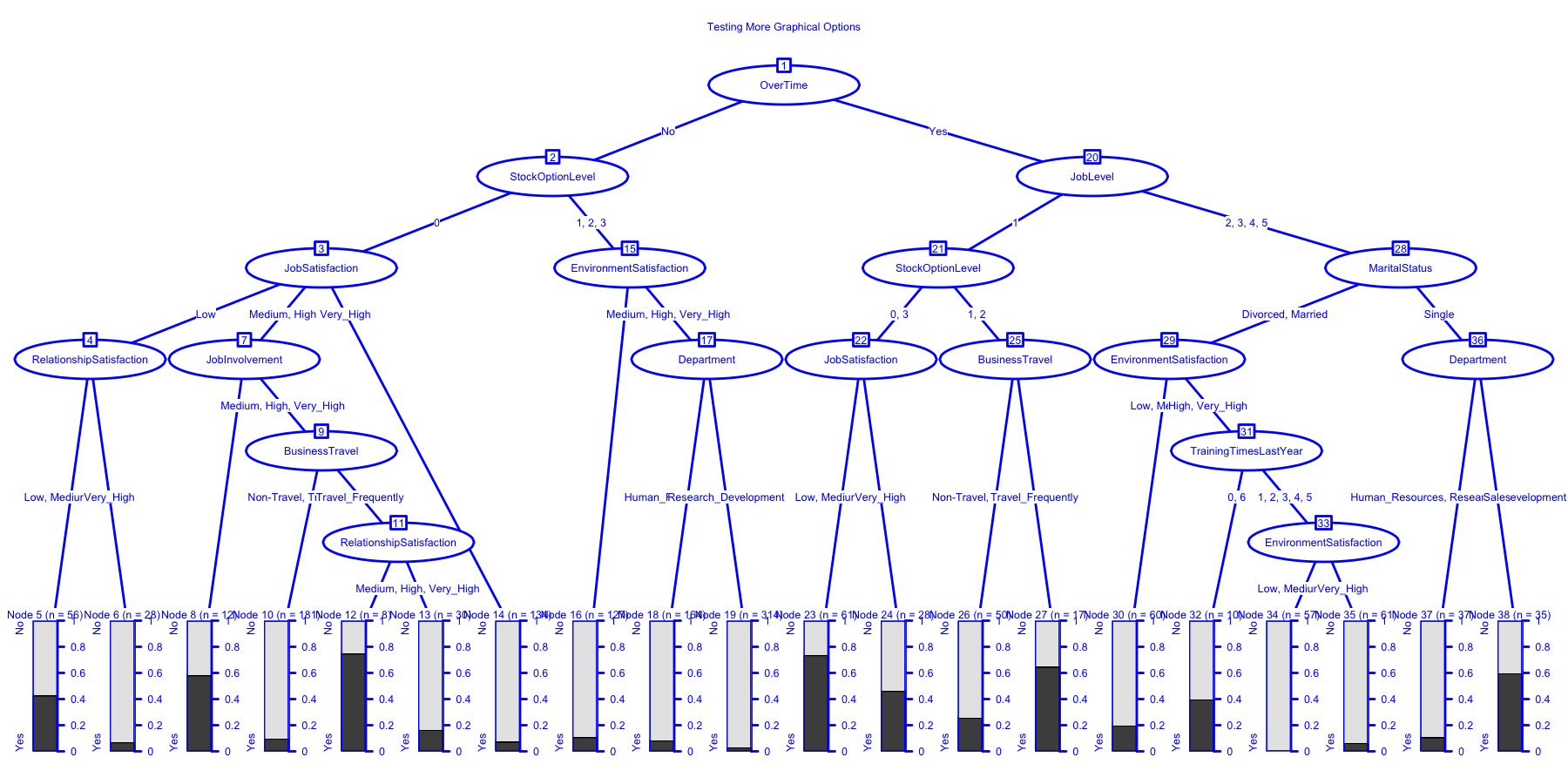
lty **=** "solid",

lwd **=** 3,

fontsize **=** 10

)

)



**Exercising some control**

Next let’s look into varying the parameters chaid uses to build the model. chaid\_control (not surprisingly) controls the behavior of the model building. When you check the documentation at ?chaid\_control you can see the list of 8 parameters you can adjust. We’ve already run the default settings implicitly when we built chaidattrit1 let’s look at three others.

* minsplit - Number of observations in splitted response at which no further split is desired.
* minprob - Minimum frequency of observations in terminal nodes.
* maxheight - Maximum height for the tree.

We’ll use those but our fourth model we’ll simply require a higher significance level for alpha2 and alpha4.

ctrl **<-** chaid\_control(minsplit **=** 200, minprob **=** 0.05)

ctrl *# notice the rest of the list is there at the default value*

## $alpha2

## [1] 0.05

##

## $alpha3

## [1] -1

##

## $alpha4

## [1] 0.05

##

## $minsplit

## [1] 200

##

## $minbucket

## [1] 7

##

## $minprob

## [1] 0.05

##

## $stump

## [1] FALSE

##

## $maxheight

## [1] -1

##

## attr(,"class")

## [1] "chaid\_control"

chaidattrit2 **<-** chaid(Attrition **~** ., data **=** newattrit, control **=** ctrl)

print(chaidattrit2)

##

## Model formula:

## Attrition ~ BusinessTravel + Department + Education + EducationField +

## EnvironmentSatisfaction + Gender + JobInvolvement + JobLevel +

## JobRole + JobSatisfaction + MaritalStatus + NumCompaniesWorked +

## OverTime + PerformanceRating + RelationshipSatisfaction +

## StockOptionLevel + TrainingTimesLastYear + WorkLifeBalance

##

## Fitted party:

## [1] root

## | [2] OverTime in No

## | | [3] StockOptionLevel in 0

## | | | [4] JobSatisfaction in Low: No (n = 84, err = 31.0%)

## | | | [5] JobSatisfaction in Medium, High

## | | | | [6] JobInvolvement in Low: Yes (n = 12, err = 41.7%)

## | | | | [7] JobInvolvement in Medium, High, Very\_High

## | | | | | [8] BusinessTravel in Non-Travel, Travel\_Rarely: No (n = 181, err = 9.9%)

## | | | | | [9] BusinessTravel in Travel\_Frequently: No (n = 38, err = 28.9%)

## | | | [10] JobSatisfaction in Very\_High: No (n = 134, err = 7.5%)

## | | [11] StockOptionLevel in 1, 2, 3

## | | | [12] EnvironmentSatisfaction in Low: No (n = 127, err = 11.0%)

## | | | [13] EnvironmentSatisfaction in Medium, High, Very\_High

## | | | | [14] Department in Human\_Resources, Sales: No (n = 164, err = 8.5%)

## | | | | [15] Department in Research\_Development: No (n = 314, err = 3.2%)

## | [16] OverTime in Yes

## | | [17] JobLevel in 1: Yes (n = 156, err = 47.4%)

## | | [18] JobLevel in 2, 3, 4, 5

## | | | [19] MaritalStatus in Divorced, Married: No (n = 188, err = 10.6%)

## | | | [20] MaritalStatus in Single: No (n = 72, err = 34.7%)

##

## Number of inner nodes: 9

## Number of terminal nodes: 11

plot(

chaidattrit2,

main **=** "minsplit = 200, minprob = 0.05",

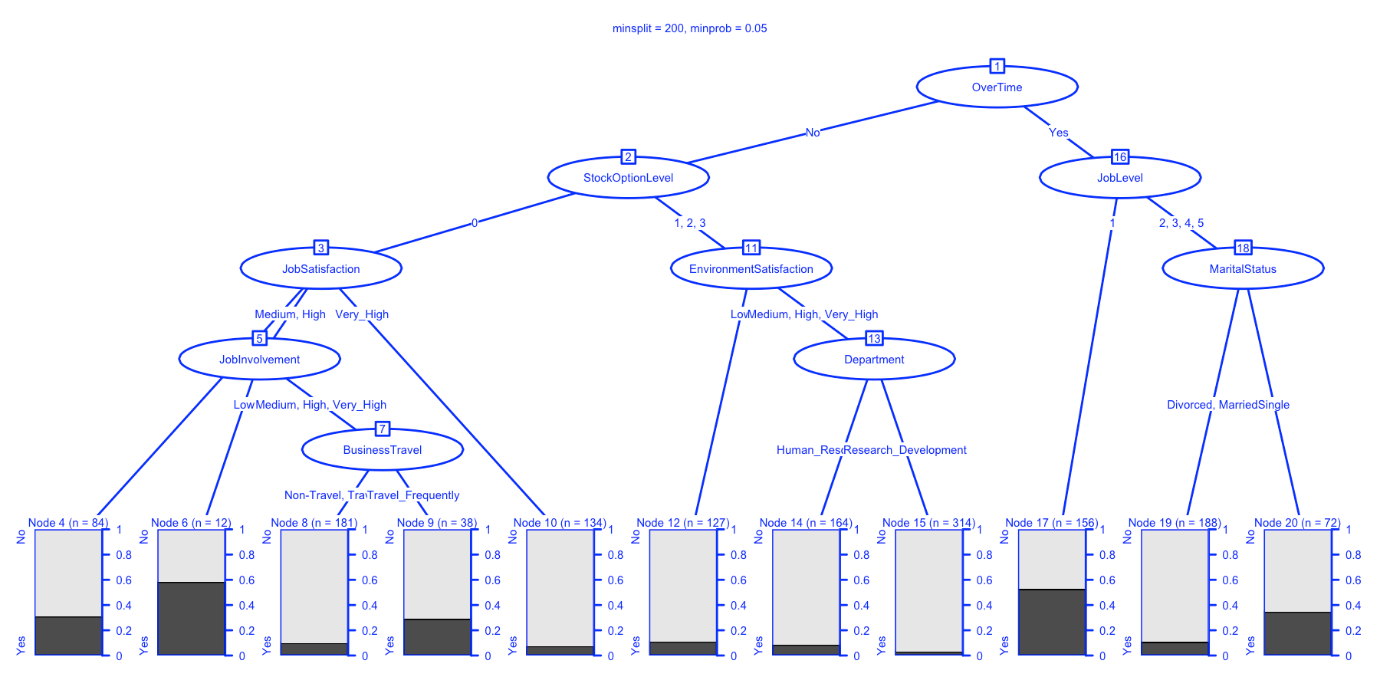
gp **=** gpar(

col **=** "blue",

lty **=** "solid",

lwd **=** 3 )

)



ctrl **<-** chaid\_control(maxheight **=** 3)

chaidattrit3 **<-** chaid(Attrition **~** ., data **=** newattrit, control **=** ctrl)

print(chaidattrit3)

##

## Model formula:

## Attrition ~ BusinessTravel + Department + Education + EducationField +

## EnvironmentSatisfaction + Gender + JobInvolvement + JobLevel +

## JobRole + JobSatisfaction + MaritalStatus + NumCompaniesWorked +

## OverTime + PerformanceRating + RelationshipSatisfaction +

## StockOptionLevel + TrainingTimesLastYear + WorkLifeBalance

##

## Fitted party:

## [1] root

## | [2] OverTime in No

## | | [3] StockOptionLevel in 0

## | | | [4] JobSatisfaction in Low: No (n = 84, err = 31.0%)

## | | | [5] JobSatisfaction in Medium, High: No (n = 231, err = 15.6%)

## | | | [6] JobSatisfaction in Very\_High: No (n = 134, err = 7.5%)

## | | [7] StockOptionLevel in 1, 2, 3

## | | | [8] EnvironmentSatisfaction in Low: No (n = 127, err = 11.0%)

## | | | [9] EnvironmentSatisfaction in Medium, High, Very\_High: No (n = 478, err = 5.0%)

## | [10] OverTime in Yes

## | | [11] JobLevel in 1

## | | | [12] StockOptionLevel in 0, 3: Yes (n = 89, err = 34.8%)

## | | | [13] StockOptionLevel in 1, 2: No (n = 67, err = 35.8%)

## | | [14] JobLevel in 2, 3, 4, 5

## | | | [15] MaritalStatus in Divorced, Married: No (n = 188, err = 10.6%)

## | | | [16] MaritalStatus in Single: No (n = 72, err = 34.7%)

##

## Number of inner nodes: 7

## Number of terminal nodes: 9

plot(

chaidattrit3,

main **=** "maxheight = 3",

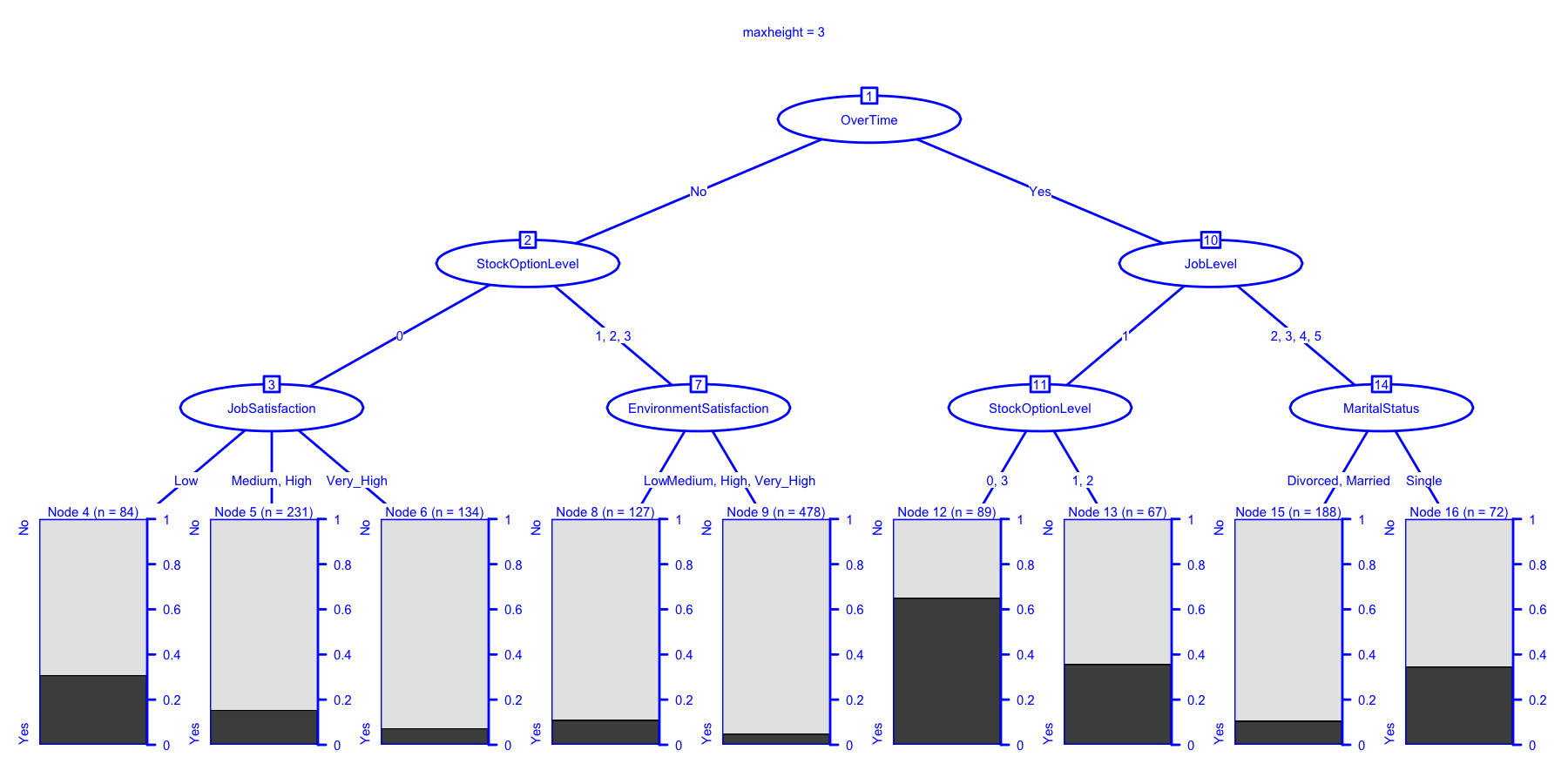
gp **=** gpar(

col **=** "blue",

lty **=** "solid",

lwd **=** 3 )

)



ctrl **<-** chaid\_control(alpha2 **=** .01, alpha4 **=** .01)

chaidattrit4 **<-** chaid(Attrition **~** ., data **=** newattrit, control **=** ctrl)

print(chaidattrit4)

##

## Model formula:

## Attrition ~ BusinessTravel + Department + Education + EducationField +

## EnvironmentSatisfaction + Gender + JobInvolvement + JobLevel +

## JobRole + JobSatisfaction + MaritalStatus + NumCompaniesWorked +

## OverTime + PerformanceRating + RelationshipSatisfaction +

## StockOptionLevel + TrainingTimesLastYear + WorkLifeBalance

##

## Fitted party:

## [1] root

## | [2] OverTime in No

## | | [3] StockOptionLevel in 0

## | | | [4] JobSatisfaction in Low

## | | | | [5] RelationshipSatisfaction in Low, Medium, High: No (n = 56, err = 42.9%)

## | | | | [6] RelationshipSatisfaction in Very\_High: No (n = 28, err = 7.1%)

## | | | [7] JobSatisfaction in Medium, High, Very\_High

## | | | | [8] JobInvolvement in Low: No (n = 20, err = 45.0%)

## | | | | [9] JobInvolvement in Medium, High, Very\_High

## | | | | | [10] JobLevel in 1: No (n = 139, err = 18.0%)

## | | | | | [11] JobLevel in 2, 3, 4, 5: No (n = 206, err = 5.8%)

## | | [12] StockOptionLevel in 1, 2, 3: No (n = 605, err = 6.3%)

## | [13] OverTime in Yes

## | | [14] JobLevel in 1

## | | | [15] StockOptionLevel in 0, 3: Yes (n = 89, err = 34.8%)

## | | | [16] StockOptionLevel in 1, 2: No (n = 67, err = 35.8%)

## | | [17] JobLevel in 2, 3, 4, 5

## | | | [18] MaritalStatus in Divorced, Married: No (n = 188, err = 10.6%)

## | | | [19] MaritalStatus in Single

## | | | | [20] Department in Human\_Resources, Research\_Development: No (n = 37, err = 10.8%)

## | | | | [21] Department in Sales: Yes (n = 35, err = 40.0%)

##

## Number of inner nodes: 10

## Number of terminal nodes: 11

plot(

chaidattrit4,

main **=** "alpha2 = .01, alpha4 = .01",

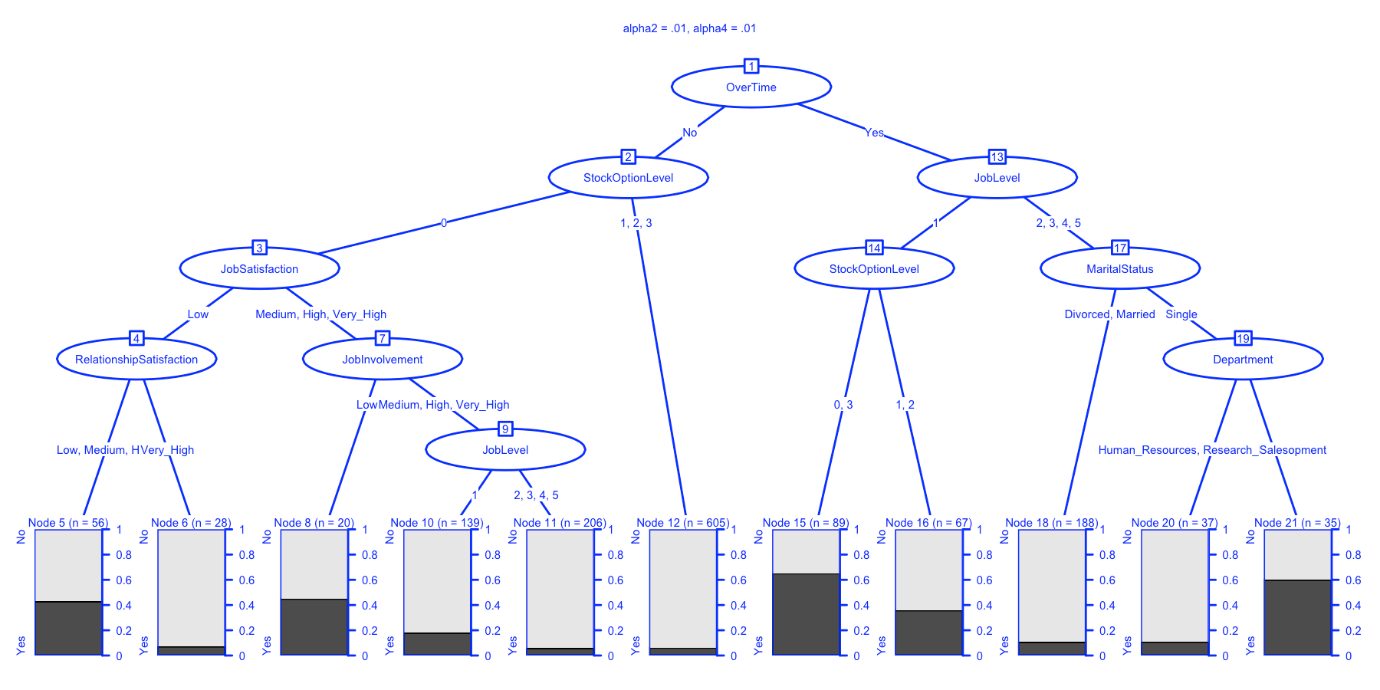
gp **=** gpar(

col **=** "blue",

lty **=** "solid",

lwd **=** 3 )

)



Let me call your attention to chaidattrit3 for a minute to highlight two important things. First it is a good picture of what we get for answer if we were to ask a question about what are the most important predictors, what variables should we focus on. An important technical detail has emerged as well. Notice that when you look at inner node #3 that there is no technical reason why a node has to have a *binary* split in chaid. As this example clearly shows node#3 leads to a three way split that is nodes #4-6.

**How good is our model?**

So the obvious question is which model is best? IMHO the joy of CHAID is in giving you a clear picture of what you would predict given the data and why. Then of course there is the usual problem every data scientist has, which is, I have what I think is a great model. How well will it generalize to new data? Whether that’s next years attrition numbers for the same company or say data from a different company.

But it’s time to talk about accuracy and all the related ideas, so on with the show…

When it’s all said and done we built a model called chaidattrit1 to be able to predict or classify the 1,470 staff members. Seems reasonable then that we can get back these predictions from the model for all 1,470 people and see how we did compared to the data we have about whether they attrited or not. The print and plot commands sort of summarize that for us at the terminal node level with an error rate but all in all which of our four models is best?

The first step is to get the predictions for each model and put them somewhere. For that we’ll use the predict command. If you inspect the object you create (in my case with a head command) you’ll see it’s a vector of factors where the attribute names is set to be the terminal node the prediction is associated with. So pmodel1 <- predict(chaidattrit1) puts our predictions using the first model we built in a nice orderly fashion. On the other side newattrit$Attrition has the actual outcome of whether the employee departed or not.

What we want is a comparison of how well we did. How often did we get it right or wrong? Turns out what we need is called a confusion matrix. The caret package has a function called confusionMatrix that will give us what we want nicely formatted and printed.

There’s a nice short summary of what is produced at this url Confusion Matrix, so I won’t even try to repeat that material. I’ll just run the appropriate commands. Later we’ll revisit this topic to be more efficient. For now I want to focus on the results.

*# digress how accurate were we*

pmodel1 **<-** predict(chaidattrit1)

head(pmodel1)

## 38 19 23 23 16 14

## Yes No Yes Yes No No

## Levels: No Yes

pmodel2 **<-** predict(chaidattrit2)

pmodel3 **<-** predict(chaidattrit3)

pmodel4 **<-** predict(chaidattrit4)

confusionMatrix(pmodel1, newattrit**$**Attrition)

## Confusion Matrix and Statistics

##

## Reference

## Prediction No Yes

## No 1190 147

## Yes 43 90

##

## Accuracy : 0.8707

## 95% CI : (0.8525, 0.8875)

## No Information Rate : 0.8388

## P-Value [Acc > NIR] : 0.0003553

##

## Kappa : 0.4192

## Mcnemar's Test P-Value : 7.874e-14

##

## Sensitivity : 0.9651

## Specificity : 0.3797

## Pos Pred Value : 0.8901

## Neg Pred Value : 0.6767

## Prevalence : 0.8388

## Detection Rate : 0.8095

## Detection Prevalence : 0.9095

## Balanced Accuracy : 0.6724

##

## 'Positive' Class : No

##

confusionMatrix(pmodel2, newattrit**$**Attrition)

## Confusion Matrix and Statistics

##

## Reference

## Prediction No Yes

## No 1154 148

## Yes 79 89

##

## Accuracy : 0.8456

## 95% CI : (0.8261, 0.8637)

## No Information Rate : 0.8388

## P-Value [Acc > NIR] : 0.2516

##

## Kappa : 0.353

## Mcnemar's Test P-Value : 6.382e-06

##

## Sensitivity : 0.9359

## Specificity : 0.3755

## Pos Pred Value : 0.8863

## Neg Pred Value : 0.5298

## Prevalence : 0.8388

## Detection Rate : 0.7850

## Detection Prevalence : 0.8857

## Balanced Accuracy : 0.6557

##

## 'Positive' Class : No

##

confusionMatrix(pmodel3, newattrit**$**Attrition)

## Confusion Matrix and Statistics

##

## Reference

## Prediction No Yes

## No 1202 179

## Yes 31 58

##

## Accuracy : 0.8571

## 95% CI : (0.8382, 0.8746)

## No Information Rate : 0.8388

## P-Value [Acc > NIR] : 0.02864

##

## Kappa : 0.2936

## Mcnemar's Test P-Value : < 2e-16

##

## Sensitivity : 0.9749

## Specificity : 0.2447

## Pos Pred Value : 0.8704

## Neg Pred Value : 0.6517

## Prevalence : 0.8388

## Detection Rate : 0.8177

## Detection Prevalence : 0.9395

## Balanced Accuracy : 0.6098

##

## 'Positive' Class : No

##

confusionMatrix(pmodel4, newattrit**$**Attrition)

## Confusion Matrix and Statistics

##

## Reference

## Prediction No Yes

## No 1188 158

## Yes 45 79

##

## Accuracy : 0.8619

## 95% CI : (0.8432, 0.8791)

## No Information Rate : 0.8388

## P-Value [Acc > NIR] : 0.007845

##

## Kappa : 0.3676

## Mcnemar's Test P-Value : 3.815e-15

##

## Sensitivity : 0.9635

## Specificity : 0.3333

## Pos Pred Value : 0.8826

## Neg Pred Value : 0.6371

## Prevalence : 0.8388

## Detection Rate : 0.8082

## Detection Prevalence : 0.9156

## Balanced Accuracy : 0.6484

##

## 'Positive' Class : No

##

There we have it, four matrices, one for each of the models we made with the different control parameters. It helpfully provides not just Accuracy but also other common measures you may be interested in. Before we leave the topic for a bit however, I do want to highlight a way you can use the purrr package to make your life a lot easier.

We have 4 models so far (with more to come) we have the nice neat output from caret but honestly to compare values across the 4 models involves way too much scrolling back and forth right now. Let’s use purrr to create a nice neat dataframe. purrr’s map command is like lapply from base R, designed to apply some operations or functions to a list of objects. So what we’ll do is as follows:

1. Create a named list called modellist to point to our four existing models (perhaps at a latter date we’ll start even earlier in our modelling process).
2. It’s a named list so we can name each model (for now with the accurate but uninteresting name Modelx)
3. Pass the list using map to the predict function to generate our predictions
4. Pipe %>% those results to the confusionMatrix function with map
5. Pipe %>% the confusion matrix results to map\_dfr. The results of confusionMattrix are actually a list of six items. The ones we want to capture are in $overall and $byClass. We grab them, transpose them, and make them into a dataframe then bind the two dataframes together so everything is neatly packaged. The .id = ModelNumb tells map\_dfr to add an identifying column to the dataframe. It is populated with the name of the list item we passed in modellist. Therefore the object CHAIDresults contains everything we might want to use to compare models in one neat dataframe.

The kable call is simply for your reading convenience. Makes it a little easier to read than a traditional print call.

library(kableExtra)

modellist **<-** **list**(Model1 **=** chaidattrit1, Model2 **=** chaidattrit2, Model3 **=** chaidattrit3, Model4 **=** chaidattrit4)

CHAIDResults **<-** map(modellist, **~** predict(.x)) **%>%**

map(**~** confusionMatrix(newattrit**$**Attrition, .x)) **%>%**

map\_dfr(**~** cbind(as.data.frame(t(.x**$**overall)),as.data.frame(t(.x**$**byClass))), .id **=** "ModelNumb")

kable(CHAIDResults, "html") **%>%**

kable\_styling(bootstrap\_options **=** **c**("striped", "hover", "condensed", "responsive"),

font\_size **=** 9)

| **ModelNumb** | **Accuracy** | **Kappa** | **AccuracyLower** | **AccuracyUpper** | **AccuracyNull** | **AccuracyPValue** | **McnemarPValue** | **Sensitivity** | **Specificity** | **Pos Pred Value** | **Neg Pred Value** | **Precision** | **Recall** | **F1** | **Prevalence** | **Detection Rate** | **Detection Prevalence** | **Balanced Accuracy** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model1 | 0.8707483 | 0.4191632 | 0.8525159 | 0.8874842 | 0.9095238 | 0.9999996 | 0.0e+00 | 0.8900524 | 0.6766917 | 0.9651257 | 0.3797468 | 0.9651257 | 0.8900524 | 0.9260700 | 0.9095238 | 0.8095238 | 0.8387755 | 0.7833720 |
| Model2 | 0.8455782 | 0.3529603 | 0.8260781 | 0.8636860 | 0.8857143 | 0.9999985 | 6.4e-06 | 0.8863287 | 0.5297619 | 0.9359286 | 0.3755274 | 0.9359286 | 0.8863287 | 0.9104536 | 0.8857143 | 0.7850340 | 0.8387755 | 0.7080453 |
| Model3 | 0.8571429 | 0.2936476 | 0.8382017 | 0.8746440 | 0.9394558 | 1.0000000 | 0.0e+00 | 0.8703838 | 0.6516854 | 0.9748581 | 0.2447257 | 0.9748581 | 0.8703838 | 0.9196634 | 0.9394558 | 0.8176871 | 0.8387755 | 0.7610346 |
| Model4 | 0.8619048 | 0.3676334 | 0.8432050 | 0.8791447 | 0.9156463 | 1.0000000 | 0.0e+00 | 0.8826152 | 0.6370968 | 0.9635036 | 0.3333333 | 0.9635036 | 0.8826152 | 0.9212873 | 0.9156463 | 0.8081633 | 0.8387755 | 0.7598560 |

One other thing I’ll mention in passing is that the partykit package offers a way of assessing the relative importance of the variables in the model via the varimp command. We’ll come back to this concept of variable importance later but for now a simple example of text and plot output.

sort(varimp(chaidattrit1), decreasing **=** **TRUE**)

## JobLevel OverTime EnvironmentSatisfaction

## 0.142756888 0.114384725 0.071069051

## StockOptionLevel MaritalStatus JobSatisfaction

## 0.058726463 0.030332565 0.029157845

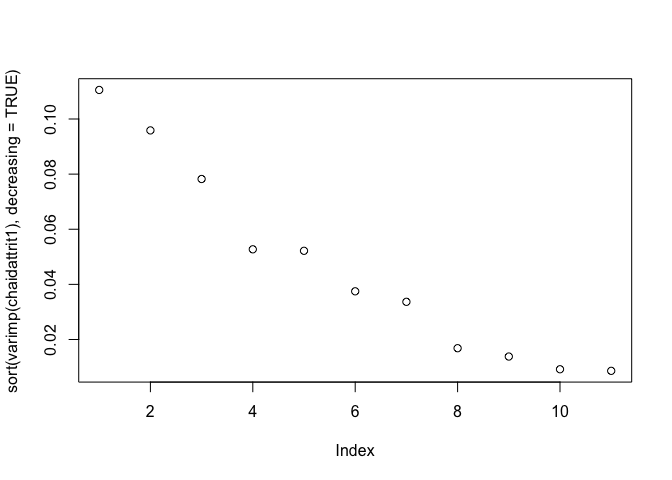
## TrainingTimesLastYear RelationshipSatisfaction Department

## 0.025637743 0.015700750 0.013815233

## BusinessTravel JobInvolvement

## 0.009906245 0.009205317

plot(sort(varimp(chaidattrit1), decreasing **=** **TRUE**))



**What about those other variables?**

But before we go much farther we should probably circle back and make use of all those variables that were coded as integers that we conveniently ignored in building our first four models. Let’s bring them into our model building activities and see what they can add to our understanding. As a first step let’s use ggplot2 and take a look at their distribution using a density plot.

*# Turning numeric variables into factors*

*## what do they look like*

attrition **%>%**

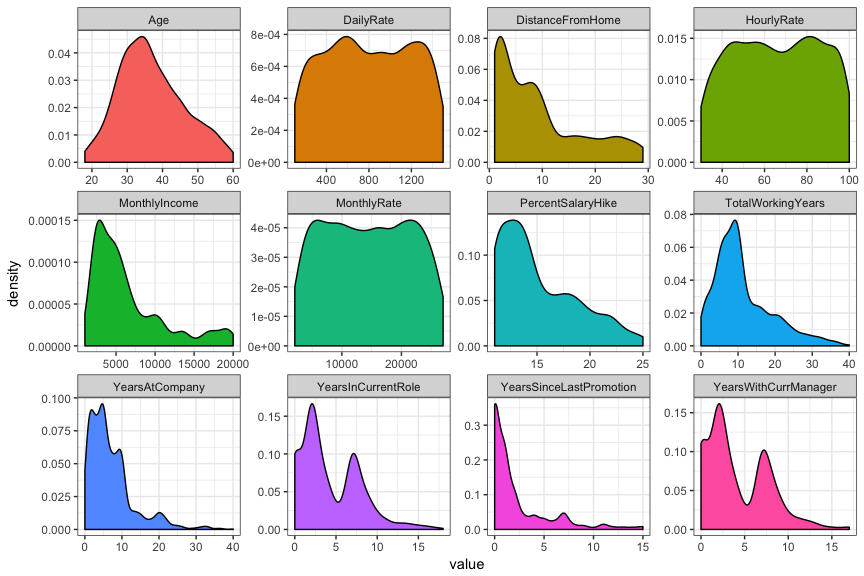
select\_if(is.numeric) **%>%**

gather(metric, value) **%>%**

ggplot(aes(value, fill **=** metric)) **+**

geom\_density(show.legend **=** **FALSE**) **+**

facet\_wrap( **~** metric, scales **=** "free")



Well other than Age very few of those variables appear to have especially normal distributions. That’s okay we’re going to wind up cutting them up into factors anyway. The only question is what are the best cut-points to use? In base R the cut function default is equal intervals (distances along the x axis). You can also specify your own cutpoints and your own labels as shown below.

table(cut(attrition**$**YearsWithCurrManager, breaks **=** 5))

##

## (-0.017,3.4] (3.4,6.8] (6.8,10.2] (10.2,13.6] (13.6,17]

## 825 158 414 54 19

table(attrition**$**YearsSinceLastPromotion)

##

## 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

## 581 357 159 52 61 45 32 76 18 17 6 24 10 10 9 13

table(cut(

attrition**$**YearsSinceLastPromotion,

breaks **=** **c**(-1, 0.9, 1.9, 2.9, 30),

labels **=** **c**("Less than 1", "1", "2", "More than 2")

))

##

## Less than 1 1 2 More than 2

## 581 357 159 373

ggplot2 has three helper functions I prefer to use: cut\_interval, cut\_number, and cut\_width. cut\_interval makes n groups with equal range, cut\_number makes n groups with (approximately) equal numbers of observations, and cut\_width makes groups of a fixed specified width. As we think about moving the numeric variables into factors any of these might be a viable alternative.

*# cut\_interval makes n groups with equal range*

table(cut\_interval(attrition**$**YearsWithCurrManager, n **=** 5))

##

## [0,3.4] (3.4,6.8] (6.8,10.2] (10.2,13.6] (13.6,17]

## 825 158 414 54 19

*# cut\_number makes n groups with (approximately) equal numbers of observations*

table(cut\_number(attrition**$**YearsWithCurrManager, n **=** 5))

##

## [0,1] (1,2] (2,4] (4,7] (7,17]

## 339 344 240 276 271

*# cut\_width makes groups of width width*

table(cut\_width(attrition**$**YearsWithCurrManager, width **=** 2))

##

## [-1,1] (1,3] (3,5] (5,7] (7,9] (9,11] (11,13] (13,15] (15,17]

## 339 486 129 245 171 49 32 10 9

For the sake of our current example let’s say that I would like to focus on groups of more or less equal size which means that I would need to apply cut\_number to each of the 12 variables under discussion. I’m not enamored of running the function 12 times though so I would prefer to wrap it in a mutate\_if statement. If the variable is numeric then apply cut\_number with n=5.

The problem is that cut\_number will error out if it doesn’t think there are enough values to produce the bins you requested. So…

cut\_number(attrition**$**YearsWithCurrManager, n **=** 6)

*# Error: Insufficient data values to produce 6 bins.*

cut\_number(attrition**$**YearsSinceLastPromotion, n **=** 4)

*# Error: Insufficient data values to produce 4 bins.*

attrition **%>%**

mutate\_if(is.numeric, funs(cut\_number(., n**=**5)))

*# Error in mutate\_impl(.data, dots) :*

*# Evaluation error: Insufficient data values to produce 5 bins..*

A little sleuthing reveals that there is one variable among the 12 that has too few values for the cut\_number function to work. That variable is YearsSinceLastPromotion. Let’s try what we would like but explicitly select out that variable.

attrition **%>%**

select(**-**YearsSinceLastPromotion) **%>%**

mutate\_if(is.numeric, funs(cut\_number(., n**=**5))) **%>%** head

## Age Attrition BusinessTravel DailyRate

## 1 (38,45] Yes Travel\_Rarely (942,1.22e+03]

## 2 (45,60] No Travel\_Frequently [102,392]

## 3 (34,38] Yes Travel\_Rarely (1.22e+03,1.5e+03]

## 4 (29,34] No Travel\_Frequently (1.22e+03,1.5e+03]

## 5 [18,29] No Travel\_Rarely (392,656]

## 6 (29,34] No Travel\_Frequently (942,1.22e+03]

## Department DistanceFromHome Education EducationField

## 1 Sales [1,2] College Life\_Sciences

## 2 Research\_Development (5,9] Below\_College Life\_Sciences

## 3 Research\_Development [1,2] College Other

## 4 Research\_Development (2,5] Master Life\_Sciences

## 5 Research\_Development [1,2] Below\_College Medical

## 6 Research\_Development [1,2] College Life\_Sciences

## EnvironmentSatisfaction Gender HourlyRate JobInvolvement JobLevel

## 1 Medium Female (87,100] High 2

## 2 High Male (59,73] Medium 2

## 3 Very\_High Male (87,100] Medium 1

## 4 Very\_High Female (45,59] High 1

## 5 Low Male [30,45] High 1

## 6 Very\_High Male (73,87] High 1

## JobRole JobSatisfaction MaritalStatus MonthlyIncome

## 1 Sales\_Executive Very\_High Single (5.74e+03,9.86e+03]

## 2 Research\_Scientist Medium Married (4.23e+03,5.74e+03]

## 3 Laboratory\_Technician High Single [1.01e+03,2.7e+03]

## 4 Research\_Scientist High Married (2.7e+03,4.23e+03]

## 5 Laboratory\_Technician Medium Married (2.7e+03,4.23e+03]

## 6 Laboratory\_Technician Very\_High Single (2.7e+03,4.23e+03]

## MonthlyRate NumCompaniesWorked OverTime PercentSalaryHike

## 1 (1.67e+04,2.17e+04] 8 Yes [11,12]

## 2 (2.17e+04,2.7e+04] 1 No (19,25]

## 3 [2.09e+03,6.89e+03] 6 Yes (13,15]

## 4 (2.17e+04,2.7e+04] 1 Yes [11,12]

## 5 (1.18e+04,1.67e+04] 9 No [11,12]

## 6 (1.18e+04,1.67e+04] 0 No (12,13]

## PerformanceRating RelationshipSatisfaction StockOptionLevel

## 1 Excellent Low 0

## 2 Outstanding Very\_High 1

## 3 Excellent Medium 0

## 4 Excellent High 0

## 5 Excellent Very\_High 1

## 6 Excellent High 0

## TotalWorkingYears TrainingTimesLastYear WorkLifeBalance YearsAtCompany

## 1 (5,8] 0 Bad (5,7]

## 2 (8,10] 3 Better (7,10]

## 3 (5,8] 3 Better [0,2]

## 4 (5,8] 3 Better (7,10]

## 5 (5,8] 3 Better [0,2]

## 6 (5,8] 2 Good (5,7]

## YearsInCurrentRole YearsWithCurrManager

## 1 (2,4] (4,7]

## 2 (4,7] (4,7]

## 3 [0,1] [0,1]

## 4 (4,7] [0,1]

## 5 (1,2] (1,2]

## 6 (4,7] (4,7]

Yes that appears to be it. So let’s manually cut it into 4 groups and then apply the 5 grouping code to the other 11 variables. Once we have accomplished that we can run the same newattrit <- attrition %>% select\_if(is.factor) we ran earlier to produce a newattrit dataframe we can work with.

attrition**$**YearsSinceLastPromotion **<-** cut(

attrition**$**YearsSinceLastPromotion,

breaks **=** **c**(-1, 0.9, 1.9, 2.9, 30),

labels **=** **c**("Less than 1", "1", "2", "More than 2")

)

attrition **<-** attrition **%>%**

mutate\_if(is.numeric, funs(cut\_number(., n**=**5)))

summary(attrition)

## Age Attrition BusinessTravel

## [18,29]:326 No :1233 Non-Travel : 150

## (29,34]:325 Yes: 237 Travel\_Frequently: 277

## (34,38]:255 Travel\_Rarely :1043

## (38,45]:291

## (45,60]:273

##

##

## DailyRate Department DistanceFromHome

## [102,392] :294 Human\_Resources : 63 [1,2] :419

## (392,656] :294 Research\_Development:961 (2,5] :213

## (656,942] :294 Sales :446 (5,9] :308

## (942,1.22e+03] :294 (9,17] :253

## (1.22e+03,1.5e+03]:294 (17,29]:277

##

##

## Education EducationField EnvironmentSatisfaction

## Below\_College:170 Human\_Resources : 27 Low :284

## College :282 Life\_Sciences :606 Medium :287

## Bachelor :572 Marketing :159 High :453

## Master :398 Medical :464 Very\_High:446

## Doctor : 48 Other : 82

## Technical\_Degree:132

##

## Gender HourlyRate JobInvolvement JobLevel

## Female:588 [30,45] :306 Low : 83 1:543

## Male :882 (45,59] :298 Medium :375 2:534

## (59,73] :280 High :868 3:218

## (73,87] :312 Very\_High:144 4:106

## (87,100]:274 5: 69

##

##

## JobRole JobSatisfaction MaritalStatus

## Sales\_Executive :326 Low :289 Divorced:327

## Research\_Scientist :292 Medium :280 Married :673

## Laboratory\_Technician :259 High :442 Single :470

## Manufacturing\_Director :145 Very\_High:459

## Healthcare\_Representative:131

## Manager :102

## (Other) :215

## MonthlyIncome MonthlyRate NumCompaniesWorked

## [1.01e+03,2.7e+03] :294 [2.09e+03,6.89e+03]:294 1 :521

## (2.7e+03,4.23e+03] :294 (6.89e+03,1.18e+04]:294 0 :197

## (4.23e+03,5.74e+03]:294 (1.18e+04,1.67e+04]:294 3 :159

## (5.74e+03,9.86e+03]:294 (1.67e+04,2.17e+04]:294 2 :146

## (9.86e+03,2e+04] :294 (2.17e+04,2.7e+04] :294 4 :139

## 7 : 74

## (Other):234

## OverTime PercentSalaryHike PerformanceRating RelationshipSatisfaction

## No :1054 [11,12]:408 Low : 0 Low :276

## Yes: 416 (12,13]:209 Good : 0 Medium :303

## (13,15]:302 Excellent :1244 High :459

## (15,19]:325 Outstanding: 226 Very\_High:432

## (19,25]:226

##

##

## StockOptionLevel TotalWorkingYears TrainingTimesLastYear WorkLifeBalance

## 0:631 [0,5] :316 0: 54 Bad : 80

## 1:596 (5,8] :309 1: 71 Good :344

## 2:158 (8,10] :298 2:547 Better:893

## 3: 85 (10,17]:261 3:491 Best :153

## (17,40]:286 4:123

## 5:119

## 6: 65

## YearsAtCompany YearsInCurrentRole YearsSinceLastPromotion

## [0,2] :342 [0,1] :301 Less than 1:581

## (2,5] :434 (1,2] :372 1 :357

## (5,7] :166 (2,4] :239 2 :159

## (7,10] :282 (4,7] :295 More than 2:373

## (10,40]:246 (7,18]:263

##

##

## YearsWithCurrManager

## [0,1] :339

## (1,2] :344

## (2,4] :240

## (4,7] :276

## (7,17]:271

##

##

newattrit **<-** attrition **%>%**

select\_if(is.factor)

**dim**(newattrit)

## [1] 1470 31

Now we have newattrit with all 30 predictor variables. We will simply repeat the process we used earlier to develop 4 new models.

*# Repeat to produce models 5-8*

chaidattrit5 **<-** chaid(Attrition **~** ., data **=** newattrit)

print(chaidattrit5)

##

## Model formula:

## Attrition ~ Age + BusinessTravel + DailyRate + Department + DistanceFromHome +

## Education + EducationField + EnvironmentSatisfaction + Gender +

## HourlyRate + JobInvolvement + JobLevel + JobRole + JobSatisfaction +

## MaritalStatus + MonthlyIncome + MonthlyRate + NumCompaniesWorked +

## OverTime + PercentSalaryHike + PerformanceRating + RelationshipSatisfaction +

## StockOptionLevel + TotalWorkingYears + TrainingTimesLastYear +

## WorkLifeBalance + YearsAtCompany + YearsInCurrentRole + YearsSinceLastPromotion +

## YearsWithCurrManager

##

## Fitted party:

## [1] root

## | [2] OverTime in No

## | | [3] YearsAtCompany in [0,2]

## | | | [4] Age in [18,29], (29,34]

## | | | | [5] StockOptionLevel in 0

## | | | | | [6] BusinessTravel in Non-Travel, Travel\_Rarely: No (n = 56, err = 41.1%)

## | | | | | [7] BusinessTravel in Travel\_Frequently: Yes (n = 10, err = 10.0%)

## | | | | [8] StockOptionLevel in 1, 2, 3: No (n = 63, err = 15.9%)

## | | | [9] Age in (34,38], (38,45], (45,60]

## | | | | [10] WorkLifeBalance in Bad: No (n = 4, err = 50.0%)

## | | | | [11] WorkLifeBalance in Good, Better, Best

## | | | | | [12] EducationField in Human\_Resources, Life\_Sciences, Marketing, Medical: No (n = 92, err = 2.2%)

## | | | | | [13] EducationField in Other, Technical\_Degree: No (n = 13, err = 23.1%)

## | | [14] YearsAtCompany in (2,5], (5,7], (7,10], (10,40]

## | | | [15] WorkLifeBalance in Bad: No (n = 45, err = 22.2%)

## | | | [16] WorkLifeBalance in Good, Better, Best

## | | | | [17] JobSatisfaction in Low

## | | | | | [18] StockOptionLevel in 0

## | | | | | | [19] RelationshipSatisfaction in Low: Yes (n = 11, err = 45.5%)

## | | | | | | [20] RelationshipSatisfaction in Medium: No (n = 12, err = 8.3%)

## | | | | | | [21] RelationshipSatisfaction in High: No (n = 17, err = 47.1%)

## | | | | | | [22] RelationshipSatisfaction in Very\_High: No (n = 20, err = 0.0%)

## | | | | | [23] StockOptionLevel in 1, 2, 3: No (n = 93, err = 4.3%)

## | | | | [24] JobSatisfaction in Medium, High, Very\_High

## | | | | | [25] Age in [18,29], (29,34], (34,38], (38,45]

## | | | | | | [26] BusinessTravel in Non-Travel, Travel\_Rarely

## | | | | | | | [27] JobInvolvement in Low: No (n = 25, err = 12.0%)

## | | | | | | | [28] JobInvolvement in Medium, High, Very\_High

## | | | | | | | | [29] RelationshipSatisfaction in Low: No (n = 81, err = 3.7%)

## | | | | | | | | [30] RelationshipSatisfaction in Medium, High: No (n = 198, err = 0.0%)

## | | | | | | | | [31] RelationshipSatisfaction in Very\_High

## | | | | | | | | | [32] DistanceFromHome in [1,2], (2,5], (5,9], (17,29]: No (n = 92, err = 2.2%)

## | | | | | | | | | [33] DistanceFromHome in (9,17]: No (n = 13, err = 23.1%)

## | | | | | | [34] BusinessTravel in Travel\_Frequently: No (n = 95, err = 8.4%)

## | | | | | [35] Age in (45,60]

## | | | | | | [36] JobSatisfaction in Low, Medium, High

## | | | | | | | [37] TotalWorkingYears in [0,5], (5,8], (8,10], (17,40]: No (n = 57, err = 0.0%)

## | | | | | | | [38] TotalWorkingYears in (10,17]: No (n = 14, err = 28.6%)

## | | | | | | [39] JobSatisfaction in Very\_High: No (n = 43, err = 20.9%)

## | [40] OverTime in Yes

## | | [41] JobLevel in 1

## | | | [42] StockOptionLevel in 0, 3

## | | | | [43] DistanceFromHome in [1,2], (2,5]

## | | | | | [44] EnvironmentSatisfaction in Low: Yes (n = 12, err = 16.7%)

## | | | | | [45] EnvironmentSatisfaction in Medium, High, Very\_High: No (n = 33, err = 36.4%)

## | | | | [46] DistanceFromHome in (5,9], (9,17], (17,29]: Yes (n = 44, err = 18.2%)

## | | | [47] StockOptionLevel in 1, 2

## | | | | [48] BusinessTravel in Non-Travel, Travel\_Rarely: No (n = 50, err = 26.0%)

## | | | | [49] BusinessTravel in Travel\_Frequently: Yes (n = 17, err = 35.3%)

## | | [50] JobLevel in 2, 3, 4, 5

## | | | [51] MaritalStatus in Divorced, Married

## | | | | [52] EnvironmentSatisfaction in Low, Medium: No (n = 60, err = 20.0%)

## | | | | [53] EnvironmentSatisfaction in High, Very\_High

## | | | | | [54] TrainingTimesLastYear in 0, 6: No (n = 10, err = 40.0%)

## | | | | | [55] TrainingTimesLastYear in 1, 2, 3, 4, 5

## | | | | | | [56] YearsInCurrentRole in [0,1], (1,2]: No (n = 36, err = 11.1%)

## | | | | | | [57] YearsInCurrentRole in (2,4], (4,7], (7,18]: No (n = 82, err = 0.0%)

## | | | [58] MaritalStatus in Single

## | | | | [59] Department in Human\_Resources, Research\_Development: No (n = 37, err = 10.8%)

## | | | | [60] Department in Sales: Yes (n = 35, err = 40.0%)

##

## Number of inner nodes: 28

## Number of terminal nodes: 32

plot(

chaidattrit5,

main **=** "Default control sliced numerics",

gp **=** gpar(

col **=** "blue",

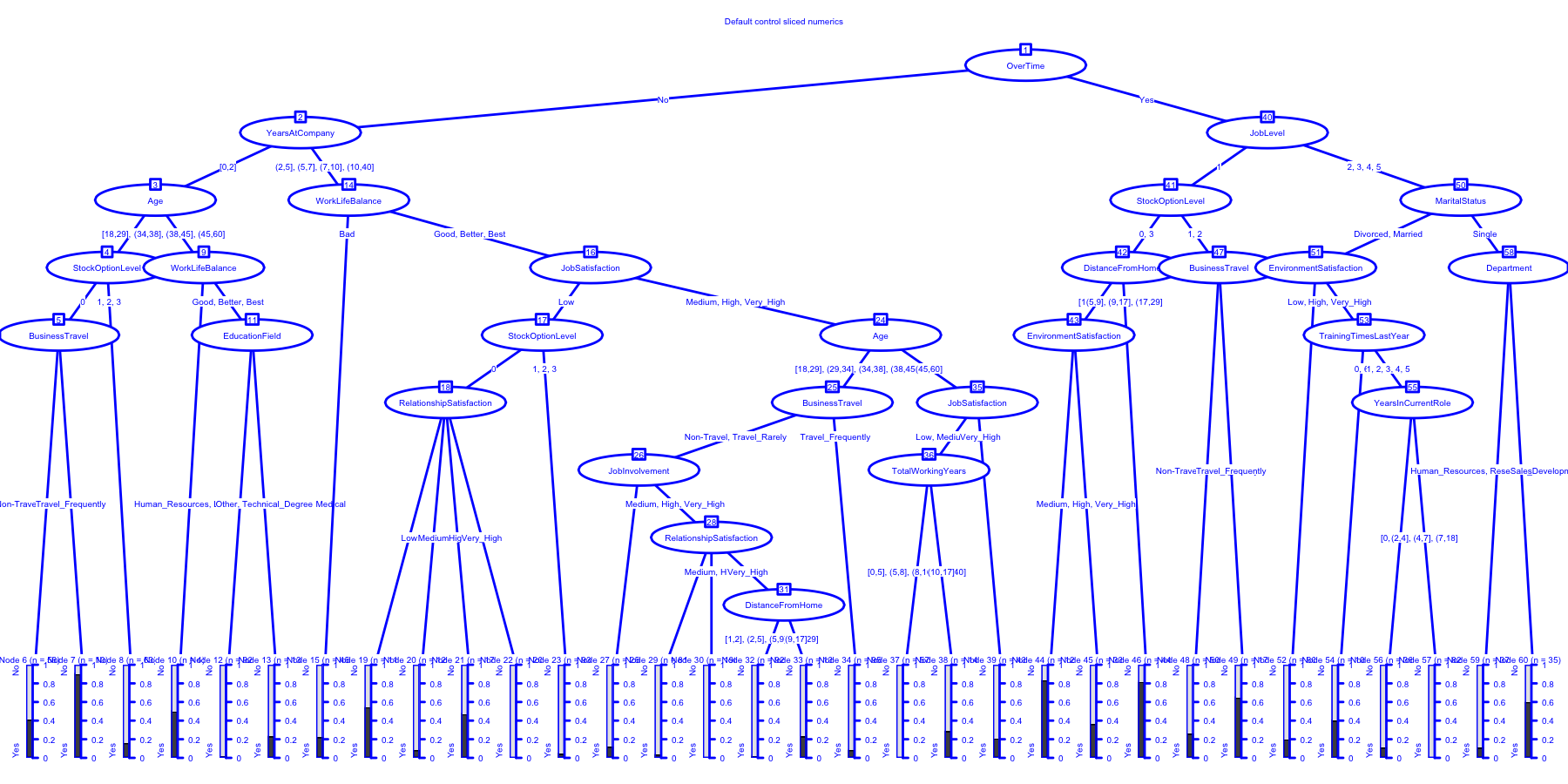
lty **=** "solid",

lwd **=** 3,

fontsize **=** 8

)

)



ctrl **<-** chaid\_control(minsplit **=** 200, minprob **=** 0.05)

chaidattrit6 **<-** chaid(Attrition **~** ., data **=** newattrit, control **=** ctrl)

print(chaidattrit6)

##

## Model formula:

## Attrition ~ Age + BusinessTravel + DailyRate + Department + DistanceFromHome +

## Education + EducationField + EnvironmentSatisfaction + Gender +

## HourlyRate + JobInvolvement + JobLevel + JobRole + JobSatisfaction +

## MaritalStatus + MonthlyIncome + MonthlyRate + NumCompaniesWorked +

## OverTime + PercentSalaryHike + PerformanceRating + RelationshipSatisfaction +

## StockOptionLevel + TotalWorkingYears + TrainingTimesLastYear +

## WorkLifeBalance + YearsAtCompany + YearsInCurrentRole + YearsSinceLastPromotion +

## YearsWithCurrManager

##

## Fitted party:

## [1] root

## | [2] OverTime in No

## | | [3] YearsAtCompany in [0,2]

## | | | [4] Age in [18,29], (29,34]: No (n = 129, err = 32.6%)

## | | | [5] Age in (34,38], (38,45], (45,60]: No (n = 109, err = 6.4%)

## | | [6] YearsAtCompany in (2,5], (5,7], (7,10], (10,40]

## | | | [7] WorkLifeBalance in Bad: No (n = 45, err = 22.2%)

## | | | [8] WorkLifeBalance in Good, Better, Best

## | | | | [9] JobSatisfaction in Low: No (n = 153, err = 12.4%)

## | | | | [10] JobSatisfaction in Medium, High, Very\_High

## | | | | | [11] Age in [18,29], (29,34], (34,38], (38,45]

## | | | | | | [12] BusinessTravel in Non-Travel, Travel\_Rarely

## | | | | | | | [13] JobInvolvement in Low: No (n = 25, err = 12.0%)

## | | | | | | | [14] JobInvolvement in Medium, High, Very\_High

## | | | | | | | | [15] RelationshipSatisfaction in Low: No (n = 81, err = 3.7%)

## | | | | | | | | [16] RelationshipSatisfaction in Medium, High: No (n = 198, err = 0.0%)

## | | | | | | | | [17] RelationshipSatisfaction in Very\_High: No (n = 105, err = 4.8%)

## | | | | | | [18] BusinessTravel in Travel\_Frequently: No (n = 95, err = 8.4%)

## | | | | | [19] Age in (45,60]: No (n = 114, err = 11.4%)

## | [20] OverTime in Yes

## | | [21] JobLevel in 1: Yes (n = 156, err = 47.4%)

## | | [22] JobLevel in 2, 3, 4, 5

## | | | [23] MaritalStatus in Divorced, Married: No (n = 188, err = 10.6%)

## | | | [24] MaritalStatus in Single: No (n = 72, err = 34.7%)

##

## Number of inner nodes: 11

## Number of terminal nodes: 13

plot(

chaidattrit6,

main **=** "minsplit = 200, minprob = 0.05",

gp **=** gpar(

col **=** "blue",

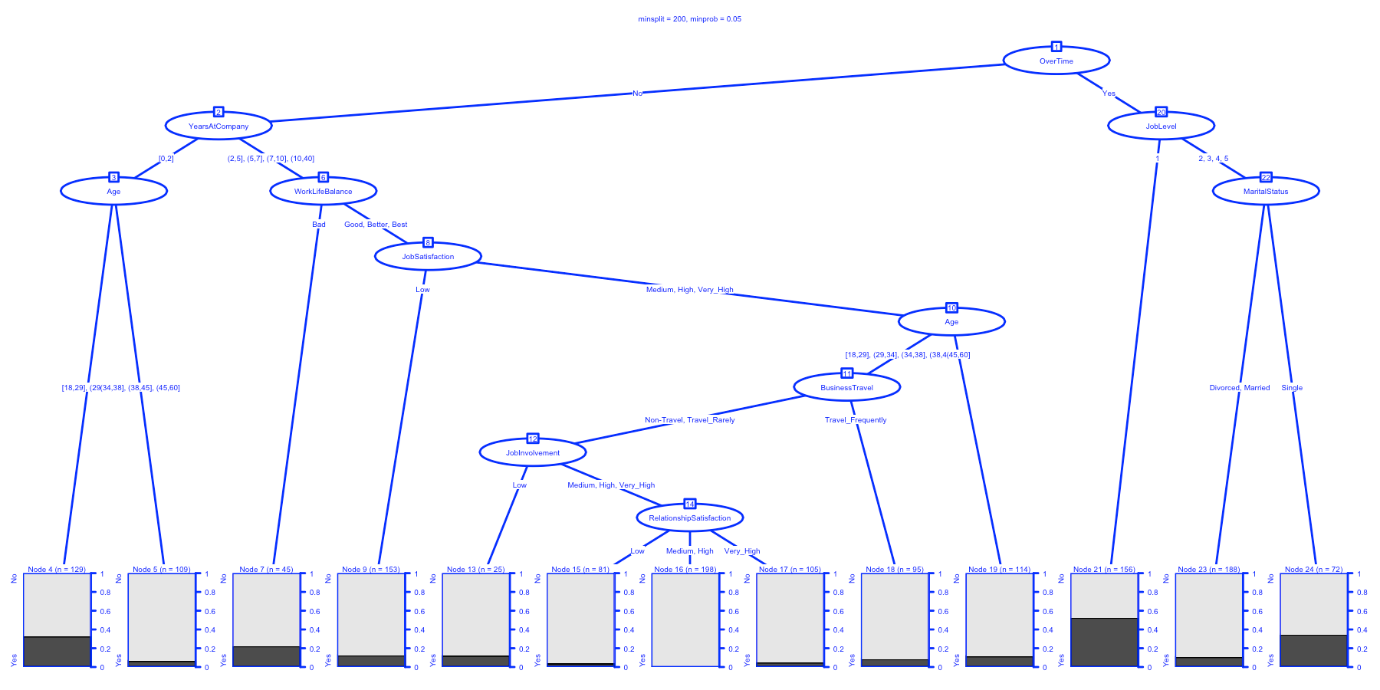
lty **=** "solid",

lwd **=** 3,

fontsize **=** 8

)

)



ctrl **<-** chaid\_control(maxheight **=** 3)

chaidattrit7 **<-** chaid(Attrition **~** ., data **=** newattrit, control **=** ctrl)

print(chaidattrit7)

##

## Model formula:

## Attrition ~ Age + BusinessTravel + DailyRate + Department + DistanceFromHome +

## Education + EducationField + EnvironmentSatisfaction + Gender +

## HourlyRate + JobInvolvement + JobLevel + JobRole + JobSatisfaction +

## MaritalStatus + MonthlyIncome + MonthlyRate + NumCompaniesWorked +

## OverTime + PercentSalaryHike + PerformanceRating + RelationshipSatisfaction +

## StockOptionLevel + TotalWorkingYears + TrainingTimesLastYear +

## WorkLifeBalance + YearsAtCompany + YearsInCurrentRole + YearsSinceLastPromotion +

## YearsWithCurrManager

##

## Fitted party:

## [1] root

## | [2] OverTime in No

## | | [3] YearsAtCompany in [0,2]

## | | | [4] Age in [18,29], (29,34]: No (n = 129, err = 32.6%)

## | | | [5] Age in (34,38], (38,45], (45,60]: No (n = 109, err = 6.4%)

## | | [6] YearsAtCompany in (2,5], (5,7], (7,10], (10,40]

## | | | [7] WorkLifeBalance in Bad: No (n = 45, err = 22.2%)

## | | | [8] WorkLifeBalance in Good, Better, Best: No (n = 771, err = 6.6%)

## | [9] OverTime in Yes

## | | [10] JobLevel in 1

## | | | [11] StockOptionLevel in 0, 3: Yes (n = 89, err = 34.8%)

## | | | [12] StockOptionLevel in 1, 2: No (n = 67, err = 35.8%)

## | | [13] JobLevel in 2, 3, 4, 5

## | | | [14] MaritalStatus in Divorced, Married: No (n = 188, err = 10.6%)

## | | | [15] MaritalStatus in Single: No (n = 72, err = 34.7%)

##

## Number of inner nodes: 7

## Number of terminal nodes: 8

plot(

chaidattrit7,

main **=** "maxheight = 3",

gp **=** gpar(

col **=** "blue",

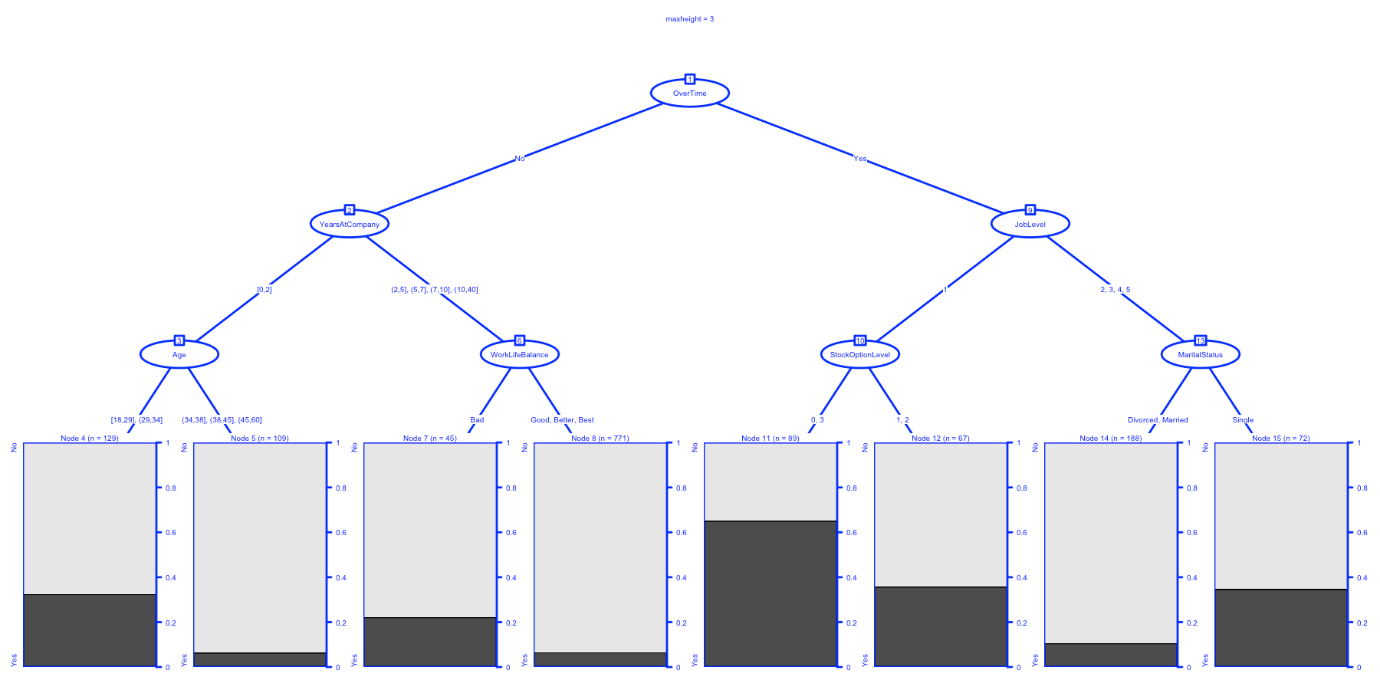
lty **=** "solid",

lwd **=** 3,

fontsize **=** 8

)

)



ctrl **<-** chaid\_control(alpha2 **=** .01, alpha4 **=** .01)

chaidattrit8 **<-** chaid(Attrition **~** ., data **=** newattrit, control **=** ctrl)

print(chaidattrit8)

##

## Model formula:

## Attrition ~ Age + BusinessTravel + DailyRate + Department + DistanceFromHome +

## Education + EducationField + EnvironmentSatisfaction + Gender +

## HourlyRate + JobInvolvement + JobLevel + JobRole + JobSatisfaction +

## MaritalStatus + MonthlyIncome + MonthlyRate + NumCompaniesWorked +

## OverTime + PercentSalaryHike + PerformanceRating + RelationshipSatisfaction +

## StockOptionLevel + TotalWorkingYears + TrainingTimesLastYear +

## WorkLifeBalance + YearsAtCompany + YearsInCurrentRole + YearsSinceLastPromotion +

## YearsWithCurrManager

##

## Fitted party:

## [1] root

## | [2] OverTime in No

## | | [3] YearsAtCompany in [0,2]

## | | | [4] Age in [18,29], (29,34]

## | | | | [5] StockOptionLevel in 0: No (n = 66, err = 48.5%)

## | | | | [6] StockOptionLevel in 1, 2, 3: No (n = 63, err = 15.9%)

## | | | [7] Age in (34,38], (38,45], (45,60]

## | | | | [8] WorkLifeBalance in Bad: No (n = 4, err = 50.0%)

## | | | | [9] WorkLifeBalance in Good, Better, Best: No (n = 105, err = 4.8%)

## | | [10] YearsAtCompany in (2,5], (5,7], (7,10], (10,40]

## | | | [11] WorkLifeBalance in Bad: No (n = 45, err = 22.2%)

## | | | [12] WorkLifeBalance in Good, Better, Best

## | | | | [13] JobSatisfaction in Low

## | | | | | [14] JobRole in Healthcare\_Representative, Human\_Resources, Laboratory\_Technician, Manager, Manufacturing\_Director, Research\_Director, Research\_Scientist, Sales\_Executive

## | | | | | | [15] StockOptionLevel in 0: No (n = 58, err = 22.4%)

## | | | | | | [16] StockOptionLevel in 1, 2, 3: No (n = 92, err = 3.3%)

## | | | | | [17] JobRole in Sales\_Representative: Yes (n = 3, err = 0.0%)

## | | | | [18] JobSatisfaction in Medium, High, Very\_High: No (n = 618, err = 5.2%)

## | [19] OverTime in Yes

## | | [20] JobLevel in 1

## | | | [21] StockOptionLevel in 0, 3: Yes (n = 89, err = 34.8%)

## | | | [22] StockOptionLevel in 1, 2: No (n = 67, err = 35.8%)

## | | [23] JobLevel in 2, 3, 4, 5

## | | | [24] MaritalStatus in Divorced, Married: No (n = 188, err = 10.6%)

## | | | [25] MaritalStatus in Single

## | | | | [26] Department in Human\_Resources, Research\_Development: No (n = 37, err = 10.8%)

## | | | | [27] Department in Sales: Yes (n = 35, err = 40.0%)

##

## Number of inner nodes: 13

## Number of terminal nodes: 14

plot(

chaidattrit8,

main **=** "alpha2 = .01, alpha4 = .01",

gp **=** gpar(

col **=** "blue",

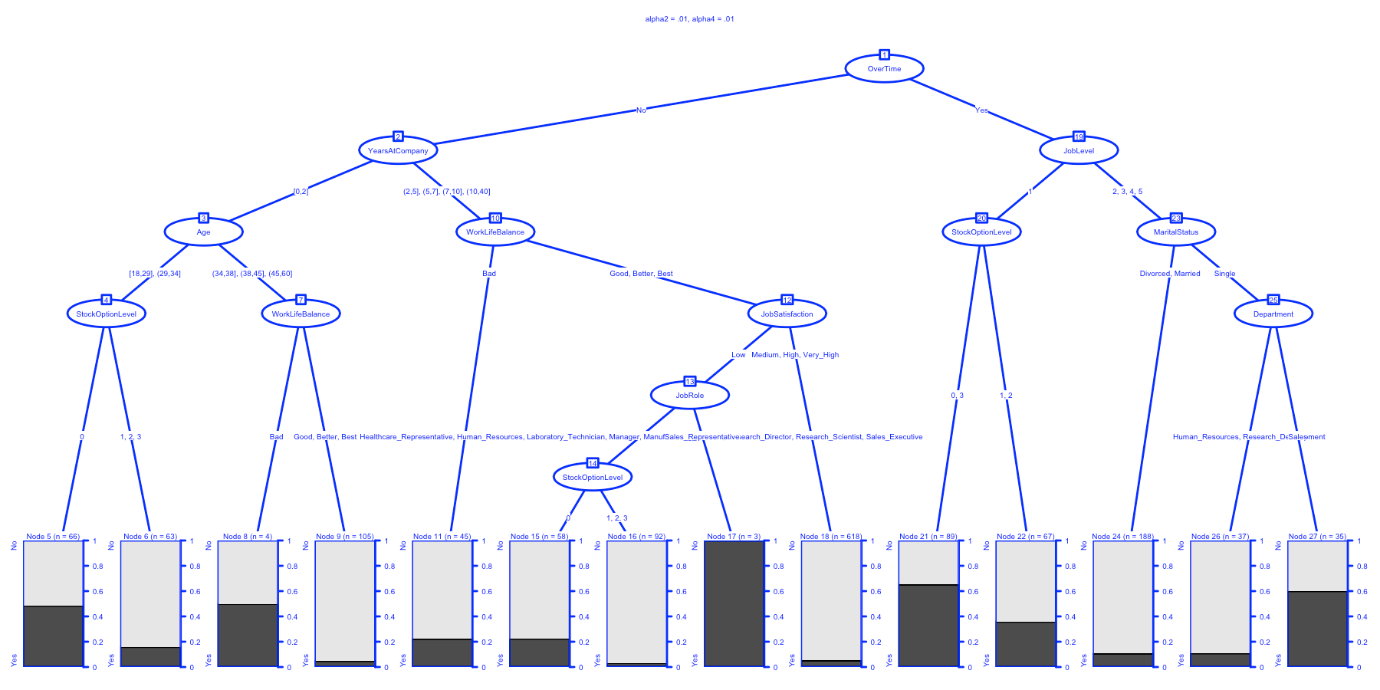
lty **=** "solid",

lwd **=** 3,

fontsize **=** 8

)

)



As we did earlier we’ll also repeat the steps necessary to build a table of results.

modellist **<-** **list**(Model1 **=** chaidattrit1,

Model2 **=** chaidattrit2,

Model3 **=** chaidattrit3,

Model4 **=** chaidattrit4,

Model5 **=** chaidattrit5,

Model6 **=** chaidattrit6,

Model7 **=** chaidattrit7,

Model8 **=** chaidattrit8)

CHAIDResults **<-** map(modellist, **~** predict(.x)) **%>%**

map(**~** confusionMatrix(newattrit**$**Attrition, .x)) **%>%**

map\_dfr(**~** cbind(as.data.frame(t(.x**$**overall)),as.data.frame(t(.x**$**byClass))), .id **=** "ModelNumb")

kable(CHAIDResults, "html") **%>%**

kable\_styling(bootstrap\_options **=** **c**("striped", "hover", "condensed", "responsive"),

font\_size **=** 10)

| **ModelNumb** | **Accuracy** | **Kappa** | **AccuracyLower** | **AccuracyUpper** | **AccuracyNull** | **AccuracyPValue** | **McnemarPValue** | **Sensitivity** | **Specificity** | **Pos Pred Value** | **Neg Pred Value** | **Precision** | **Recall** | **F1** | **Prevalence** | **Detection Rate** | **Detection Prevalence** | **Balanced Accuracy** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model1 | 0.8707483 | 0.4191632 | 0.8525159 | 0.8874842 | 0.9095238 | 0.9999996 | 0.0e+00 | 0.8900524 | 0.6766917 | 0.9651257 | 0.3797468 | 0.9651257 | 0.8900524 | 0.9260700 | 0.9095238 | 0.8095238 | 0.8387755 | 0.7833720 |
| Model2 | 0.8455782 | 0.3529603 | 0.8260781 | 0.8636860 | 0.8857143 | 0.9999985 | 6.4e-06 | 0.8863287 | 0.5297619 | 0.9359286 | 0.3755274 | 0.9359286 | 0.8863287 | 0.9104536 | 0.8857143 | 0.7850340 | 0.8387755 | 0.7080453 |
| Model3 | 0.8571429 | 0.2936476 | 0.8382017 | 0.8746440 | 0.9394558 | 1.0000000 | 0.0e+00 | 0.8703838 | 0.6516854 | 0.9748581 | 0.2447257 | 0.9748581 | 0.8703838 | 0.9196634 | 0.9394558 | 0.8176871 | 0.8387755 | 0.7610346 |
| Model4 | 0.8619048 | 0.3676334 | 0.8432050 | 0.8791447 | 0.9156463 | 1.0000000 | 0.0e+00 | 0.8826152 | 0.6370968 | 0.9635036 | 0.3333333 | 0.9635036 | 0.8826152 | 0.9212873 | 0.9156463 | 0.8081633 | 0.8387755 | 0.7598560 |
| Model5 | 0.8775510 | 0.4451365 | 0.8596959 | 0.8938814 | 0.9122449 | 0.9999968 | 0.0e+00 | 0.8926174 | 0.7209302 | 0.9708029 | 0.3924051 | 0.9708029 | 0.8926174 | 0.9300699 | 0.9122449 | 0.8142857 | 0.8387755 | 0.8067738 |
| Model6 | 0.8442177 | 0.3317731 | 0.8246542 | 0.8623944 | 0.8938776 | 1.0000000 | 1.0e-07 | 0.8820396 | 0.5256410 | 0.9399838 | 0.3459916 | 0.9399838 | 0.8820396 | 0.9100903 | 0.8938776 | 0.7884354 | 0.8387755 | 0.7038403 |
| Model7 | 0.8571429 | 0.2936476 | 0.8382017 | 0.8746440 | 0.9394558 | 1.0000000 | 0.0e+00 | 0.8703838 | 0.6516854 | 0.9748581 | 0.2447257 | 0.9748581 | 0.8703838 | 0.9196634 | 0.9394558 | 0.8176871 | 0.8387755 | 0.7610346 |
| Model8 | 0.8639456 | 0.3808988 | 0.8453515 | 0.8810715 | 0.9136054 | 1.0000000 | 0.0e+00 | 0.8845867 | 0.6456693 | 0.9635036 | 0.3459916 | 0.9635036 | 0.8845867 | 0.9223602 | 0.9136054 | 0.8081633 | 0.8387755 | 0.7651280 |

You can clearly see that Overtime remains the first cut in our tree structure but that now other variables have started to influence our model as well, such as how long they’ve worked for us and their age. You can see from the table that model #5 is apparently the most accurate now. Not by a huge amount but apparently these numeric variables we ignored at first pass do matter at least to some degree.

# explore the control options

ctrl <- chaid\_control(minsplit = 200, minprob = 0.05)

ctrl

## $alpha2

## [1] 0.05

##

## $alpha3

## [1] -1

##

## $alpha4

## [1] 0.05

##

## $minsplit

## [1] 200

##

## $minbucket

## [1] 7

##

## $minprob

## [1] 0.05

##

## $stump

## [1] FALSE

##

## $maxheight

## [1] -1

##

## attr(,"class")

## [1] "chaid\_control"

full\_data <- chaid(Attrition ~ ., data = newattrit, control = ctrl)

print(full\_data)

##

## Model formula:

## Attrition ~ Age + BusinessTravel + DailyRate + Department + DistanceFromHome +

## Education + EducationField + EnvironmentSatisfaction + Gender +

## HourlyRate + JobInvolvement + JobLevel + JobRole + JobSatisfaction +

## MaritalStatus + MonthlyIncome + MonthlyRate + NumCompaniesWorked +

## OverTime + PercentSalaryHike + PerformanceRating + RelationshipSatisfaction +

## StockOptionLevel + TotalWorkingYears + TrainingTimesLastYear +

## WorkLifeBalance + YearsAtCompany + YearsInCurrentRole + YearsSinceLastPromotion +

## YearsWithCurrManager

##

## Fitted party:

## [1] root

## | [2] OverTime in No

## | | [3] YearsAtCompany in [0,2]

## | | | [4] Age in [18,29], (29,34]: No (n = 129, err = 32.6%)

## | | | [5] Age in (34,38], (38,45], (45,60]: No (n = 109, err = 6.4%)

## | | [6] YearsAtCompany in (2,5], (5,7], (7,10], (10,40]

## | | | [7] WorkLifeBalance in Bad: No (n = 45, err = 22.2%)

## | | | [8] WorkLifeBalance in Good, Better, Best

## | | | | [9] JobSatisfaction in Low: No (n = 153, err = 12.4%)

## | | | | [10] JobSatisfaction in Medium, High, Very\_High

## | | | | | [11] Age in [18,29], (29,34], (34,38], (38,45]

## | | | | | | [12] BusinessTravel in Non-Travel, Travel\_Rarely

## | | | | | | | [13] JobInvolvement in Low: No (n = 25, err = 12.0%)

## | | | | | | | [14] JobInvolvement in Medium, High, Very\_High

## | | | | | | | | [15] RelationshipSatisfaction in Low: No (n = 81, err = 3.7%)

## | | | | | | | | [16] RelationshipSatisfaction in Medium, High: No (n = 198, err = 0.0%)

## | | | | | | | | [17] RelationshipSatisfaction in Very\_High: No (n = 105, err = 4.8%)

## | | | | | | [18] BusinessTravel in Travel\_Frequently: No (n = 95, err = 8.4%)

## | | | | | [19] Age in (45,60]: No (n = 114, err = 11.4%)

## | [20] OverTime in Yes

## | | [21] JobLevel in 1: Yes (n = 156, err = 47.4%)

## | | [22] JobLevel in 2, 3, 4, 5

## | | | [23] MaritalStatus in Divorced, Married: No (n = 188, err = 10.6%)

## | | | [24] MaritalStatus in Single: No (n = 72, err = 34.7%)

##

## Number of inner nodes: 11

## Number of terminal nodes: 13

plot(

full\_data,

main = "newattrit dataset, minsplit = 200, minprob = 0.05",

gp = gpar(

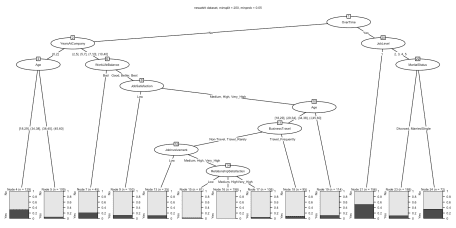
lty = "solid",

lwd = 2,

fontsize = 10

)

)



**Over-fitting**

Okay we have a working predictive model. At this point, however, we’ve  
been **cheating** to a certain degree! We’ve been using every available  
piece of data we have to develop the best possible model. We’ve told the  
powerful all-knowing algorithims to squeeze every last bit of accuracy  
they can out of the data. We’ve told it to fit the best possible  
model. Problem is that we may have done that at the cost of being able  
to generalize our model to new data or to new situations. That’s the  
problem of over-fitting in a nutshell. I’m going to move on to a solution for solving this limitation and  
that’s where caret comes in.

As a first step, let’s just take 30% of our data and put is aside for a  
minute. We’re not going to let chaid *see it* or know about it as we  
build the model. In some scenarios you have subsequent data at hand for  
checking your model (data from another company or another year or …). We  
don’t, so we’re going to self-impose this restraint. Why 30%? Doesn’t  
have to be, could be as low as 20% or as high as 40% it really depends  
on how conservative you want to be, and how much data you have at hand.  
Since this is just a tutorial we’ll simply use 30% as a representative  
number. We’ve already loaded both rsample and caret either of which  
is quite capable of making this split for us. I’m arbitrarily going to  
use rsample syntax which is the line with initial\_split(newattrit,  
prop = .7, strata = "Attrition") in it. That takes our data set  
newattrit makes a 70% split ensuring that we keep our outcome variable  
Attrition as close to 70/30 as we can. *This is important because our  
data is already pretty lop-sided* for outcomes. The two subsequent lines  
serve to take the data contained in split and produce two separate  
dataframes, test and train. They have 440 and 1030 staff members  
each. We’ll set test aside for now and focus on train.

# Create training (70%) and test (30%) sets for the attrition data.

# Use set.seed for reproducibility

#####

set.seed(1234)

split <- initial\_split(newattrit, prop = .7, strata = "Attrition")

train <- training(split)

test <- testing(split)

The next step is a little counter-intuitive but quite practical. Turns  
out that many models do not perform well when you feed them a formula  
for the model even if they claim to support a formula interface (as  
CHAID does). We’re just taking our  
predictors or features and putting them in x while we put our  
outcome in y.

Library(CHAID)

# create response and feature data

features <- setdiff(names(train), "Attrition")

x <- train[, features]

y <- train$Attrition

Alright, let’s get back on track. trainControl is the function within  
caret we need to use. Chapter 5 in the caret doco covers it in great  
detail. I’m simply going to pluck out a few sane and safe options.  
method = "cv" gets us cross-validation. number = 10 is pretty  
obvious. I happen to like seeing the progress in case I want to go for  
coffee so verboseIter = TRUE, and I play it safe and explicitly save  
my predictions savePredictions = "final". We put everything in  
train\_control which we’ll use in a minute.

# set up 10-fold cross validation procedure

train\_control <- trainControl(method = "cv",

number = 10,

verboseIter = TRUE,

savePredictions = "final")

Not surprisingly the train function in caret trains our model! It  
wants to know what our x and y’s are, as well as our training  
control parameters which we’ve parked in train\_control. At this point  
we could successfully unleash the dogs of war (sorry Shakespeare) and  
train our model since we know we want to use chaid. But let’s change  
one other useful thing and that is metric which is what metric we want  
to use to pick the “best” model. Instead of the default “accuracy” we’ll  
use Kappa which as you may remember from the last post is more  
conservative measure of how well we did.

**If you’re running this code yourself this is a good time to take a  
coffee break. I’ll tell you later how to find out how long it took  
more or less exactly. But there’s no getting around it we’re model  
building many more times so it takes longer.**

# train model

chaid.m1 <- train(

x = x,

y = y,

method = "chaid",

metric = "Kappa",

trControl = train\_control

)

## + Fold01: alpha2=0.05, alpha3=-1, alpha4=0.05

## - Fold01: alpha2=0.05, alpha3=-1, alpha4=0.05

## + Fold01: alpha2=0.03, alpha3=-1, alpha4=0.03

## - Fold01: alpha2=0.03, alpha3=-1, alpha4=0.03

## + Fold01: alpha2=0.01, alpha3=-1, alpha4=0.01

## - Fold01: alpha2=0.01, alpha3=-1, alpha4=0.01

## + Fold02: alpha2=0.05, alpha3=-1, alpha4=0.05

## - Fold02: alpha2=0.05, alpha3=-1, alpha4=0.05

## + Fold02: alpha2=0.03, alpha3=-1, alpha4=0.03

## - Fold02: alpha2=0.03, alpha3=-1, alpha4=0.03

## + Fold02: alpha2=0.01, alpha3=-1, alpha4=0.01

## - Fold02: alpha2=0.01, alpha3=-1, alpha4=0.01

## + Fold03: alpha2=0.05, alpha3=-1, alpha4=0.05

## - Fold03: alpha2=0.05, alpha3=-1, alpha4=0.05

## + Fold03: alpha2=0.03, alpha3=-1, alpha4=0.03

## - Fold03: alpha2=0.03, alpha3=-1, alpha4=0.03

## + Fold03: alpha2=0.01, alpha3=-1, alpha4=0.01

## - Fold03: alpha2=0.01, alpha3=-1, alpha4=0.01

## + Fold04: alpha2=0.05, alpha3=-1, alpha4=0.05

## - Fold04: alpha2=0.05, alpha3=-1, alpha4=0.05

## + Fold04: alpha2=0.03, alpha3=-1, alpha4=0.03

## - Fold04: alpha2=0.03, alpha3=-1, alpha4=0.03

## + Fold04: alpha2=0.01, alpha3=-1, alpha4=0.01

## - Fold04: alpha2=0.01, alpha3=-1, alpha4=0.01

## + Fold05: alpha2=0.05, alpha3=-1, alpha4=0.05

## - Fold05: alpha2=0.05, alpha3=-1, alpha4=0.05

## + Fold05: alpha2=0.03, alpha3=-1, alpha4=0.03

## - Fold05: alpha2=0.03, alpha3=-1, alpha4=0.03

## + Fold05: alpha2=0.01, alpha3=-1, alpha4=0.01

## - Fold05: alpha2=0.01, alpha3=-1, alpha4=0.01

## + Fold06: alpha2=0.05, alpha3=-1, alpha4=0.05

## - Fold06: alpha2=0.05, alpha3=-1, alpha4=0.05

## + Fold06: alpha2=0.03, alpha3=-1, alpha4=0.03

## - Fold06: alpha2=0.03, alpha3=-1, alpha4=0.03

## + Fold06: alpha2=0.01, alpha3=-1, alpha4=0.01

## - Fold06: alpha2=0.01, alpha3=-1, alpha4=0.01

## + Fold07: alpha2=0.05, alpha3=-1, alpha4=0.05

## - Fold07: alpha2=0.05, alpha3=-1, alpha4=0.05

## + Fold07: alpha2=0.03, alpha3=-1, alpha4=0.03

## - Fold07: alpha2=0.03, alpha3=-1, alpha4=0.03

## + Fold07: alpha2=0.01, alpha3=-1, alpha4=0.01

## - Fold07: alpha2=0.01, alpha3=-1, alpha4=0.01

## + Fold08: alpha2=0.05, alpha3=-1, alpha4=0.05

## - Fold08: alpha2=0.05, alpha3=-1, alpha4=0.05

## + Fold08: alpha2=0.03, alpha3=-1, alpha4=0.03

## - Fold08: alpha2=0.03, alpha3=-1, alpha4=0.03

## + Fold08: alpha2=0.01, alpha3=-1, alpha4=0.01

## - Fold08: alpha2=0.01, alpha3=-1, alpha4=0.01

## + Fold09: alpha2=0.05, alpha3=-1, alpha4=0.05

## - Fold09: alpha2=0.05, alpha3=-1, alpha4=0.05

## + Fold09: alpha2=0.03, alpha3=-1, alpha4=0.03

## - Fold09: alpha2=0.03, alpha3=-1, alpha4=0.03

## + Fold09: alpha2=0.01, alpha3=-1, alpha4=0.01

## - Fold09: alpha2=0.01, alpha3=-1, alpha4=0.01

## + Fold10: alpha2=0.05, alpha3=-1, alpha4=0.05

## - Fold10: alpha2=0.05, alpha3=-1, alpha4=0.05

## + Fold10: alpha2=0.03, alpha3=-1, alpha4=0.03

## - Fold10: alpha2=0.03, alpha3=-1, alpha4=0.03

## + Fold10: alpha2=0.01, alpha3=-1, alpha4=0.01

## - Fold10: alpha2=0.01, alpha3=-1, alpha4=0.01

## Aggregating results

## Selecting tuning parameters

## Fitting alpha2 = 0.05, alpha3 = -1, alpha4 = 0.05 on full training set

And…. we’re done. Turns out in this case the best solution was what  
chaid uses as defaults. The very last line of the output tells us  
that. But let’s use what we have used in the past for printing and  
plotting the results…

chaid.m1 #equivalent to print(chaid.m1)

## CHi-squared Automated Interaction Detection

##

## 1030 samples

## 30 predictor

## 2 classes: 'No', 'Yes'

##

## No pre-processing

## Resampling: Cross-Validated (10 fold)

## Summary of sample sizes: 928, 927, 927, 926, 928, 926, ...

## Resampling results across tuning parameters:

##

## alpha2 alpha4 Accuracy Kappa

## 0.01 0.01 0.8223292 0.1522392

## 0.03 0.03 0.8349699 0.1579585

## 0.05 0.05 0.8213958 0.1692826

##

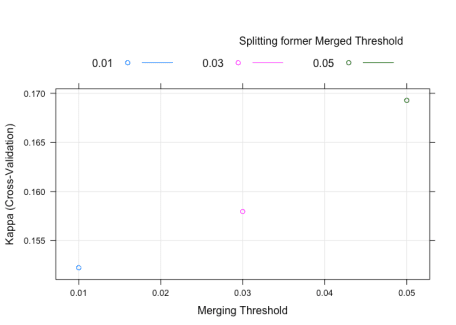
## Tuning parameter 'alpha3' was held constant at a value of -1

## Kappa was used to select the optimal model using the largest value.

## The final values used for the model were alpha2 = 0.05, alpha3 = -1

## and alpha4 = 0.05.

plot(chaid.m1)



Wait. What? These are not the output we’re used to. caret has changed  
the output from its’ work (an improvement actually) but we’ll have to  
change how we get the information out. Before we do that however, let’s  
inspect what we have so far. The output gives us a nice concise summary.  
1030 cases with 30 predictors. It gives us an idea of how many of the  
1030 cases were used in the individual folds Summary of sample  
sizes: 928, 927, 927, 926, 928, 926, ....

The bit about alpha2, alpha4, and alpha3 is somewhat mysterious.  
We saw those names when we looked at the chaid\_control documentation  
last post but why are they here? We’ll come back to that in a moment.  
But it is clear that it thought Kappa of 0.1692826 was best.

The plot isn’t what we’re used to seeing, but is easy to understand.  
Kappa is on the y axis, alpha2 on the x axis and it’s shaded/colored  
by alpha4 (remember we left alpha3 out of the mix). The plot is a  
bit of overkill for what we did but we’ll put it to better use later.

But what about the things we were used to seeing? Well if you remember  
that caret is reporting averages of all the folds it sort of makes  
sense that the **best** final model results are now in  
chaid.m1$finalModel so we need to use that when we print or plot.  
So in the next block of code let’s:

1. Print the final model from chaid (chaid.m1$finalModel)
2. Plot the final model from chaid (plot(chaid.m1$finalModel))
3. Produce the confusionMatrix across all folds  
   (confusionMatrix(chaid.m1))
4. Produce the confusionMatrix using the final model  
   (confusionMatrix(predict(chaid.m1), y))
5. Check on variable importance (varImp(chaid.m1))
6. The best tuning parameters are stored in chaid.m1$bestTune
7. How long did it take? Look in chaid.m1$times
8. In case you forgot what method you used look here chaid.m1$method
9. We’ll look at model info in a bit chaid.m1$modelInfo
10. The summarized results are here in a nice format if needed later  
    chaid.m1$results

Many of these you’ll never need but I wanted to at least give you a hint  
of how complete the chaid.m1 object is

chaid.m1$finalModel

##

## Model formula:

## .outcome ~ Age + BusinessTravel + DailyRate + Department + DistanceFromHome +

## Education + EducationField + EnvironmentSatisfaction + Gender +

## HourlyRate + JobInvolvement + JobLevel + JobRole + JobSatisfaction +

## MaritalStatus + MonthlyIncome + MonthlyRate + NumCompaniesWorked +

## OverTime + PercentSalaryHike + PerformanceRating + RelationshipSatisfaction +

## StockOptionLevel + TotalWorkingYears + TrainingTimesLastYear +

## WorkLifeBalance + YearsAtCompany + YearsInCurrentRole + YearsSinceLastPromotion +

## YearsWithCurrManager

##

## Fitted party:

## [1] root

## | [2] OverTime in No

## | | [3] YearsAtCompany in [0,2]

## | | | [4] Age in [18,29], (29,34]

## | | | | [5] StockOptionLevel in 0: No (n = 43, err = 48.8%)

## | | | | [6] StockOptionLevel in 1, 2, 3

## | | | | | [7] RelationshipSatisfaction in Low: Yes (n = 7, err = 42.9%)

## | | | | | [8] RelationshipSatisfaction in Medium, High, Very\_High: No (n = 38, err = 7.9%)

## | | | [9] Age in (34,38], (38,45], (45,60]: No (n = 77, err = 7.8%)

## | | [10] YearsAtCompany in (2,5], (5,7], (7,10], (10,40]

## | | | [11] WorkLifeBalance in Bad: No (n = 36, err = 19.4%)

## | | | [12] WorkLifeBalance in Good, Better, Best

## | | | | [13] Department in Human\_Resources, Sales

## | | | | | [14] Age in [18,29], (29,34], (34,38], (38,45]

## | | | | | | [15] WorkLifeBalance in Bad, Good: No (n = 37, err = 16.2%)

## | | | | | | [16] WorkLifeBalance in Better, Best: No (n = 119, err = 4.2%)

## | | | | | [17] Age in (45,60]: No (n = 27, err = 25.9%)

## | | | | [18] Department in Research\_Development: No (n = 347, err = 4.0%)

## | [19] OverTime in Yes

## | | [20] JobLevel in 1

## | | | [21] JobRole in Healthcare\_Representative, Human\_Resources, Laboratory\_Technician, Manager, Manufacturing\_Director, Research\_Director, Sales\_Executive, Sales\_Representative

## | | | | [22] JobInvolvement in Low, Medium: Yes (n = 19, err = 10.5%)

## | | | | [23] JobInvolvement in High, Very\_High: Yes (n = 45, err = 44.4%)

## | | | [24] JobRole in Research\_Scientist: No (n = 53, err = 35.8%)

## | | [25] JobLevel in 2, 3, 4, 5

## | | | [26] Gender in Female: No (n = 86, err = 9.3%)

## | | | [27] Gender in Male

## | | | | [28] MaritalStatus in Divorced, Married: No (n = 71, err = 18.3%)

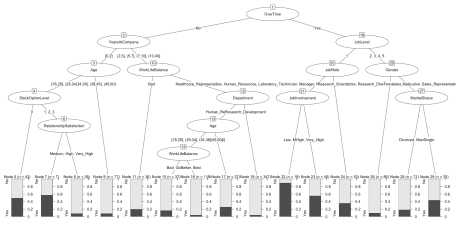
## | | | | [29] MaritalStatus in Single: No (n = 25, err = 44.0%)

##

## Number of inner nodes: 14

## Number of terminal nodes: 15

plot(chaid.m1$finalModel)



confusionMatrix(chaid.m1)

## Cross-Validated (10 fold) Confusion Matrix

##

## (entries are percentual average cell counts across resamples)

##

## Reference

## Prediction No Yes

## No 79.0 13.0

## Yes 4.9 3.1

##

## Accuracy (average) : 0.8214

confusionMatrix(predict(chaid.m1), y)

## Confusion Matrix and Statistics

##

## Reference

## Prediction No Yes

## No 839 120

## Yes 25 46

##

## Accuracy : 0.8592

## 95% CI : (0.8365, 0.8799)

## No Information Rate : 0.8388

## P-Value [Acc > NIR] : 0.03938

##

## Kappa : 0.3228

## Mcnemar's Test P-Value : 5.89e-15

##

## Sensitivity : 0.9711

## Specificity : 0.2771

## Pos Pred Value : 0.8749

## Neg Pred Value : 0.6479

## Prevalence : 0.8388

## Detection Rate : 0.8146

## Detection Prevalence : 0.9311

## Balanced Accuracy : 0.6241

##

## 'Positive' Class : No

##

varImp(chaid.m1)

## ROC curve variable importance

##

## only 20 most important variables shown (out of 30)

##

## Importance

## OverTime 100.00

## YearsInCurrentRole 90.81

## YearsAtCompany 90.41

## MonthlyIncome 87.08

## JobLevel 84.36

## TotalWorkingYears 80.04

## YearsWithCurrManager 79.78

## StockOptionLevel 69.51

## MaritalStatus 65.96

## Age 59.31

## JobSatisfaction 44.86

## JobInvolvement 44.27

## DistanceFromHome 36.80

## EnvironmentSatisfaction 32.15

## WorkLifeBalance 31.63

## DailyRate 30.23

## JobRole 29.94

## NumCompaniesWorked 28.67

## Department 25.79

## HourlyRate 19.81

chaid.m1$bestTune

## alpha2 alpha3 alpha4

## 3 0.05 -1 0.05

chaid.m1$times

## $everything

## user system elapsed

## 247.218 1.581 248.999

##

## $final

## user system elapsed

## 9.612 0.055 9.674

##

## $prediction

## [1] NA NA NA

chaid.m1$method

## [1] "chaid"

chaid.m1$modelInfo

## $label

## [1] "CHi-squared Automated Interaction Detection"

##

## $library

## [1] "CHAID"

##

## $loop

## NULL

##

## $type

## [1] "Classification"

##

## $parameters

## parameter class

## 1 alpha2 numeric

## 2 alpha3 numeric

## 3 alpha4 numeric

## label

## 1 Merging Threshold

## 2 Splitting former Merged Threshold

## 3 \n Splitting former Merged Threshold

##

## $grid

## function (x, y, len = NULL, search = "grid")

## {

## if (search == "grid") {

## out <- data.frame(alpha2 = seq(from = 0.05, to = 0.01,

## length = len), alpha3 = -1, alpha4 = seq(from = 0.05,

## to = 0.01, length = len))

## }

## else {

## out <- data.frame(alpha2 = runif(len, min = 1e-06, max = 0.1),

## alpha3 = runif(len, min = -0.1, max = 0.1), alpha4 = runif(len,

## min = 1e-06, max = 0.1))

## }

## out

## }

##

## $fit

## function (x, y, wts, param, lev, last, classProbs, ...)

## {

## dat <- if (is.data.frame(x))

## x

## else as.data.frame(x)

## dat$.outcome <- y

## theDots <- list(...)

## if (any(names(theDots) == "control")) {

## theDots$control$alpha2 <- param$alpha2

## theDots$control$alpha3 <- param$alpha3

## theDots$control$alpha4 <- param$alpha4

## ctl <- theDots$control

## theDots$control <- NULL

## }

## else ctl <- chaid\_control(alpha2 = param$alpha2, alpha3 = param$alpha3,

## alpha4 = param$alpha4)

## if (!is.null(wts))

## theDots$weights <- wts

## modelArgs <- c(list(formula = as.formula(".outcome ~ ."),

## data = dat, control = ctl), theDots)

## out <- do.call(CHAID::chaid, modelArgs)

## out

## }

##

##

## $predict

## function (modelFit, newdata, submodels = NULL)

## {

## if (!is.data.frame(newdata))

## newdata <- as.data.frame(newdata)

## predict(modelFit, newdata)

## }

##

##

## $prob

## function (modelFit, newdata, submodels = NULL)

## {

## if (!is.data.frame(newdata))

## newdata <- as.data.frame(newdata)

## predict(modelFit, newdata, type = "prob")

## }

##

## $levels

## function (x)

## x$obsLevels

##

## $predictors

## function (x, surrogate = TRUE, ...)

## {

## predictors(terms(x))

## }

##

## $tags

## [1] "Tree-Based Model" "Implicit Feature Selection"

## [3] "Two Class Only" "Accepts Case Weights"

##

## $sort

## function (x)

## x[order(-x$alpha2, -x$alpha4, -x$alpha3), ]

chaid.m1$results

## alpha2 alpha3 alpha4 Accuracy Kappa AccuracySD KappaSD

## 1 0.01 -1 0.01 0.8223292 0.1522392 0.01887938 0.1278739

## 2 0.03 -1 0.03 0.8349699 0.1579585 0.02503052 0.1093852

## 3 0.05 -1 0.05 0.8213958 0.1692826 0.03353654 0.1180522

**Let’s tune it up a little**

Having mastered the basics of using caret and chaid let’s explore a  
little deeper. By default caret allows us to adjust three parameters  
in our chaid model; alpha2, alpha3, and alpha4. As a matter of  
fact it will allow us to build a grid of those parameters and test all  
the permutations we like, using the same cross-validation process. I’m a  
bit worried that we’re not being conservative enough. I’d like to train  
our model using p values for alpha that are not .05, .03, and .01 but  
instead the de facto levels in my discipline; .05, .01, and .001. The  
function in caret is tuneGrid. We’ll use the base R function  
expand.grid to build a dataframe with all the combinations and then  
feed it to caret in our next training.

Therefore search\_grid will hold the values and we’ll add the line  
tuneGrid = search\_grid to our call to train. We’ll call the results  
chaid.m2 and see how we did (I’m turning off verbose iteration output  
since you’ve seen it on screen once already)…

# set up tuning grid default

search\_grid <- expand.grid(

alpha2 = c(.05, .01, .001),

alpha4 = c(.05, .01, .001),

alpha3 = -1

)

# no verbose

train\_control <- trainControl(method = "cv",

number = 10,

savePredictions = "final")

# train model

chaid.m2 <- train(

x = x,

y = y,

method = "chaid",

metric = "Kappa",

trControl = train\_control,

tuneGrid = search\_grid

)

chaid.m2

## CHi-squared Automated Interaction Detection

##

## 1030 samples

## 30 predictor

## 2 classes: 'No', 'Yes'

##

## No pre-processing

## Resampling: Cross-Validated (10 fold)

## Summary of sample sizes: 926, 927, 928, 928, 928, 926, ...

## Resampling results across tuning parameters:

##

## alpha2 alpha4 Accuracy Kappa

## 0.001 0.001 0.8378522 0.2755221

## 0.001 0.010 0.8329691 0.2039261

## 0.001 0.050 0.8231655 0.2026735

## 0.010 0.001 0.8378522 0.2755221

## 0.010 0.010 0.8358914 0.2185542

## 0.010 0.050 0.8280863 0.2231160

## 0.050 0.001 0.8407648 0.2992935

## 0.050 0.010 0.8387949 0.2487845

## 0.050 0.050 0.8280296 0.2324447

##

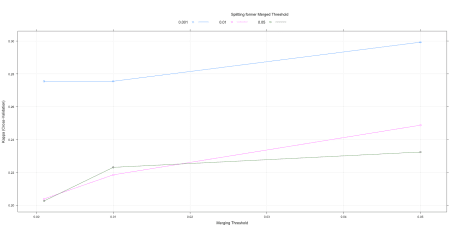
## Tuning parameter 'alpha3' was held constant at a value of -1

## Kappa was used to select the optimal model using the largest value.

## The final values used for the model were alpha2 = 0.05, alpha3 = -1

## and alpha4 = 0.001.

plot(chaid.m2)



chaid.m2$finalModel

##

## Model formula:

## .outcome ~ Age + BusinessTravel + DailyRate + Department + DistanceFromHome +

## Education + EducationField + EnvironmentSatisfaction + Gender +

## HourlyRate + JobInvolvement + JobLevel + JobRole + JobSatisfaction +

## MaritalStatus + MonthlyIncome + MonthlyRate + NumCompaniesWorked +

## OverTime + PercentSalaryHike + PerformanceRating + RelationshipSatisfaction +

## StockOptionLevel + TotalWorkingYears + TrainingTimesLastYear +

## WorkLifeBalance + YearsAtCompany + YearsInCurrentRole + YearsSinceLastPromotion +

## YearsWithCurrManager

##

## Fitted party:

## [1] root

## | [2] OverTime in No

## | | [3] YearsAtCompany in [0,2]: No (n = 165, err = 20.6%)

## | | [4] YearsAtCompany in (2,5], (5,7], (7,10], (10,40]: No (n = 566, err = 6.9%)

## | [5] OverTime in Yes

## | | [6] JobLevel in 1: Yes (n = 117, err = 47.9%)

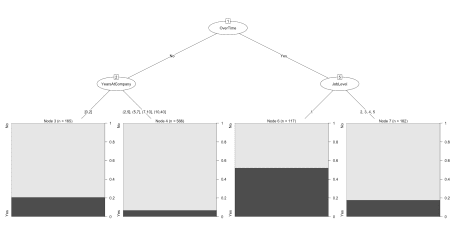
## | | [7] JobLevel in 2, 3, 4, 5: No (n = 182, err = 17.6%)

##

## Number of inner nodes: 3

## Number of terminal nodes: 4

plot(chaid.m2$finalModel)



confusionMatrix(chaid.m2)

## Cross-Validated (10 fold) Confusion Matrix

##

## (entries are percentual average cell counts across resamples)

##

## Reference

## Prediction No Yes

## No 79.0 11.1

## Yes 4.9 5.0

##

## Accuracy (average) : 0.8408

confusionMatrix(predict(chaid.m2), y)

## Confusion Matrix and Statistics

##

## Reference

## Prediction No Yes

## No 808 105

## Yes 56 61

##

## Accuracy : 0.8437

## 95% CI : (0.82, 0.8653)

## No Information Rate : 0.8388

## P-Value [Acc > NIR] : 0.354533

##

## Kappa : 0.3436

## Mcnemar's Test P-Value : 0.000155

##

## Sensitivity : 0.9352

## Specificity : 0.3675

## Pos Pred Value : 0.8850

## Neg Pred Value : 0.5214

## Prevalence : 0.8388

## Detection Rate : 0.7845

## Detection Prevalence : 0.8864

## Balanced Accuracy : 0.6513

##

## 'Positive' Class : No

##

chaid.m2$times

## $everything

## user system elapsed

## 524.972 3.729 529.873

##

## $final

## user system elapsed

## 2.173 0.013 2.191

##

## $prediction

## [1] NA NA NA

chaid.m2$results

## alpha2 alpha4 alpha3 Accuracy Kappa AccuracySD KappaSD

## 1 0.001 0.001 -1 0.8378522 0.2755221 0.02253555 0.09552095

## 2 0.001 0.010 -1 0.8329691 0.2039261 0.02263752 0.09977861

## 3 0.001 0.050 -1 0.8231655 0.2026735 0.03187552 0.12676157

## 4 0.010 0.001 -1 0.8378522 0.2755221 0.02253555 0.09552095

## 5 0.010 0.010 -1 0.8358914 0.2185542 0.02240334 0.10717030

## 6 0.010 0.050 -1 0.8280863 0.2231160 0.03056971 0.08137926

## 7 0.050 0.001 -1 0.8407648 0.2992935 0.02523390 0.10729121

## 8 0.050 0.010 -1 0.8387949 0.2487845 0.02277103 0.10696016

## 9 0.050 0.050 -1 0.8280296 0.2324447 0.03157911 0.13890292

Very nice! Some key points here. Even though our model got more  
conservative and has far fewer nodes, our accuracy has improved as  
measured both by traditional accuracy and Kappa. That applies at both  
the average fold level but more importantly at the *best model*  
prediction stage. Later on we’ll start using our models to predict  
against the data we held out in test.

The plot is also more useful now. No matter what we do with alpha2 it  
pays to keep alpha4 conservative at .001 (blue line always on top) but  
keeping alpha2 modest seems to be best.

This goes to the heart of our conversation about over-fitting. While it  
may seem like 1,400+ cases is a lot of data we are at great risk of  
over-fitting if we try and build too complex a model, so sometimes a  
conservative track is warranted.

**A Custom caret model**

Earlier I printed the results of chaid.m1$modelInfo and then pretty  
much skipped over discussing them. Under the covers one of the strengths  
of caret is that it keeps some default information about how to tune  
various types of algorithms.

Code Chunks - CHAID.R

|  |
| --- |
| modelInfo <- list(label = "CHi-squared Automated Interaction Detection", |
|  | library = "CHAID", |
|  | loop = NULL, |
|  | type = c("Classification"), |
|  | parameters = data.frame(parameter = c('alpha2', 'alpha3', 'alpha4'), |
|  | class = rep('numeric', 3), |
|  | label = c('Merging Threshold', |
|  | "Splitting former Merged Threshold", " |
|  | Splitting former Merged Threshold")), |
|  | grid = function(x, y, len = NULL, search = "grid") { |
|  | if(search == "grid") { |
|  | out <- data.frame(alpha2 = seq(from = .05, to = 0.01, length = len), |
|  | alpha3 = -1, |
|  | alpha4 = seq(from = .05, to = 0.01, length = len)) |
|  | } else { |
|  | out <- data.frame(alpha2 = runif(len, min = 0.000001, max = .1), |
|  | alpha3 = runif(len, min =-.1, max = .1), |
|  | alpha4 = runif(len, min = 0.000001, max = .1)) |
|  | } |
|  | out |
|  | }, |
|  | fit = function(x, y, wts, param, lev, last, classProbs, ...) { |
|  | dat <- if(is.data.frame(x)) x else as.data.frame(x, stringsAsFactors = TRUE) |
|  | dat$.outcome <- y |
|  | theDots <- list(...) |
|  | if(any(names(theDots) == "control")) { |
|  | theDots$control$alpha2 <- param$alpha2 |
|  | theDots$control$alpha3 <- param$alpha3 |
|  | theDots$control$alpha4 <- param$alpha4 |
|  | ctl <- theDots$control |
|  | theDots$control <- NULL |
|  | } else ctl <- CHAID::chaid\_control(alpha2 = param$alpha2, |
|  | alpha3 = param$alpha3, |
|  | alpha4 = param$alpha4) |
|  | ## pass in any model weights |
|  | if(!is.null(wts)) theDots$weights <- wts |
|  | modelArgs <- c( |
|  | list( |
|  | formula = as.formula(".outcome ~ ."), |
|  | data = dat, |
|  | control = ctl), |
|  | theDots) |
|  | out <- do.call(CHAID::chaid, modelArgs) |
|  | out |
|  | }, |
|  | predict = function(modelFit, newdata, submodels = NULL) { |
|  | if(!is.data.frame(newdata)) newdata <- as.data.frame(newdata, stringsAsFactors = TRUE) |
|  | predict(modelFit, newdata) |
|  | }, |
|  | prob = function(modelFit, newdata, submodels = NULL) { |
|  | if(!is.data.frame(newdata)) newdata <- as.data.frame(newdata, stringsAsFactors = TRUE) |
|  | predict(modelFit, newdata, type = "prob") |
|  | }, |
|  | levels = function(x) x$obsLevels, |
|  | predictors = function(x, surrogate = TRUE, ...) { |
|  | predictors(terms(x)) |
|  | }, |
|  | tags = c('Tree-Based Model', "Implicit Feature Selection", "Two Class Only", "Accepts Case Weights"), |
|  | sort = function(x) x[order(-x$alpha2, -x$alpha4, -x$alpha3),]) |

My experience is that they are quite comprehensive and allow you to get  
your modelling done. But sometimes you want to do something your own way  
or different and caret has provisions for that. That is not a comprehensive list of all the  
parameters we can work with in chaid\_control see ?chaid\_control for  
a listing and brief description of what they all are.

What if, for example, we wanted to tune based upon minsplit,  
minbucket, minprob, maxheight instead? How would we go about using  
all the built in functionality in caret but have it our way? At first it  
looked a little too complicated for my tastes, but I found that with a  
bit of trial and error I was able to hack up the existing list that I  
found on GITHUB and convert it into a list in my local environment that  
worked perfectly for my needs.

I won’t bore you with all the details and the documentation is quite  
good so it wound up being mainly a search and replace operation and  
adding one parameter. I decided to call my version cgpCHAID and here’s  
what the version looks like.

# hack up my own

Library(CHAID)

Library(CARET)

cgpCHAID <- list(label = "CGP CHAID",

library = "CHAID",

loop = NULL,

type = c("Classification"),

parameters = data.frame(parameter = c('minsplit', 'minbucket', 'minprob', 'maxheight'),

class = rep('numeric', 4),

label = c('Numb obs in response where no further split',

"Minimum numb obs in terminal nodes",

"Minimum freq of obs in terminal nodes.",

"Maximum height for the tree")

),

grid = function(x, y, len = NULL, search = "grid") {

if(search == "grid") {

out <- data.frame(minsplit = c(20,30),

minbucket = 7,

minprob = c(0.05,0.01),

maxheight = -1)

} else {

out <- data.frame(minsplit = c(20,30),

minbucket = 7,

minprob = c(0.05,0.01),

maxheight = -1)

}

out

},

fit = function(x, y, wts, param, lev, last, classProbs, ...) {

dat <- if(is.data.frame(x)) x else as.data.frame(x)

dat$.outcome <- y

theDots <- list(...)

if(any(names(theDots) == "control")) {

theDots$control$minsplit <- param$minsplit

theDots$control$minbucket <- param$minbucket

theDots$control$minprob <- param$minprob

theDots$control$maxheight <- param$maxheight

ctl <- theDots$control

theDots$control <- NULL

} else ctl <- chaid\_control(minsplit = param$minsplit,

minbucket = param$minbucket,

minprob = param$minprob,

maxheight = param$maxheight)

## pass in any model weights

if(!is.null(wts)) theDots$weights <- wts

modelArgs <- c(

list(

formula = as.formula(".outcome ~ ."),

data = dat,

control = ctl),

theDots)

out <- do.call(CHAID::chaid, modelArgs)

out

},

predict = function(modelFit, newdata, submodels = NULL) {

if(!is.data.frame(newdata)) newdata <- as.data.frame(newdata)

predict(modelFit, newdata)

},

prob = function(modelFit, newdata, submodels = NULL) {

if(!is.data.frame(newdata)) newdata <- as.data.frame(newdata)

predict(modelFit, newdata, type = "prob")

},

levels = function(x) x$obsLevels,

predictors = function(x, surrogate = TRUE, ...) {

predictors(terms(x))

},

tags = c('Tree-Based Model', "Implicit Feature Selection", "Two Class Only", "Accepts Case Weights"),

sort = function(x) x[order(-x$minsplit, -x$minbucket, -x$minprob, -x$maxheight),])

cgpCHAID

## $label

## [1] "CGP CHAID"

##

## $library

## [1] "CHAID"

##

## $loop

## NULL

##

## $type

## [1] "Classification"

##

## $parameters

## parameter class label

## 1 minsplit numeric Numb obs in response where no further split

## 2 minbucket numeric Minimum numb obs in terminal nodes

## 3 minprob numeric Minimum freq of obs in terminal nodes.

## 4 maxheight numeric Maximum height for the tree

##

## $grid

## function (x, y, len = NULL, search = "grid")

## {

## if (search == "grid") {

## out <- data.frame(minsplit = c(20, 30), minbucket = 7,

## minprob = c(0.05, 0.01), maxheight = -1)

## }

## else {

## out <- data.frame(minsplit = c(20, 30), minbucket = 7,

## minprob = c(0.05, 0.01), maxheight = -1)

## }

## out

## }

##

## $fit

## function (x, y, wts, param, lev, last, classProbs, ...)

## {

## dat <- if (is.data.frame(x))

## x

## else as.data.frame(x)

## dat$.outcome <- y

## theDots <- list(...)

## if (any(names(theDots) == "control")) {

## theDots$control$minsplit <- param$minsplit

## theDots$control$minbucket <- param$minbucket

## theDots$control$minprob <- param$minprob

## theDots$control$maxheight <- param$maxheight

## ctl <- theDots$control

## theDots$control <- NULL

## }

## else ctl <- chaid\_control(minsplit = param$minsplit, minbucket = param$minbucket,

## minprob = param$minprob, maxheight = param$maxheight)

## if (!is.null(wts))

## theDots$weights <- wts

## modelArgs <- c(list(formula = as.formula(".outcome ~ ."),

## data = dat, control = ctl), theDots)

## out <- do.call(CHAID::chaid, modelArgs)

## out

## }

##

## $predict

## function (modelFit, newdata, submodels = NULL)

## {

## if (!is.data.frame(newdata))

## newdata <- as.data.frame(newdata)

## predict(modelFit, newdata)

## }

##

## $prob

## function (modelFit, newdata, submodels = NULL)

## {

## if (!is.data.frame(newdata))

## newdata <- as.data.frame(newdata)

## predict(modelFit, newdata, type = "prob")

## }

##

## $levels

## function (x)

## x$obsLevels

##

## $predictors

## function (x, surrogate = TRUE, ...)

## {

## predictors(terms(x))

## }

##

## $tags

## [1] "Tree-Based Model" "Implicit Feature Selection"

## [3] "Two Class Only" "Accepts Case Weights"

##

## $sort

## function (x)

## x[order(-x$minsplit, -x$minbucket, -x$minprob, -x$maxheight),

## ]

The final print statement shows what it looks like and confirms it is  
there ready for us to use in the local environment. The original chaid  
version in caret remains untouched and available in caret for when  
we want it. To make use of our custom model we simply rebuild our search  
grid using our new parameters.

# set up tuning grid cgpCHAID

search\_grid <- expand.grid(

minsplit = c(30,40),

minprob = .1,

minbucket = 25,

maxheight = 4

)

search\_grid

## minsplit minprob minbucket maxheight

## 1 30 0.1 25 4

## 2 40 0.1 25 4

Then to use it to train our third model chaid.m3 we insert it into the  
method directive (**not quoted** because it’s in the local  
environment).

# train model

chaid.m3 <- train(

x = x,

y = y,

method = cgpCHAID,

trControl = train\_control,

metric = "Kappa",

tuneGrid = search\_grid

)

The process runs for a few minutes and then produces output very similar  
to what we received for chaid.m2. We get summarized information across  
our 10 folds and the all important The final values used for the model  
were minsplit = 40, minbucket = 25, minprob = 0.1 and maxheight = 4. I  
won’t review all the details since I’ve already covered it I’ve simply  
printed it out to confirm it all works.

chaid.m3

## CGP CHAID

##

## 1030 samples

## 30 predictor

## 2 classes: 'No', 'Yes'

##

## No pre-processing

## Resampling: Cross-Validated (10 fold)

## Summary of sample sizes: 926, 927, 927, 927, 927, 927, ...

## Resampling results across tuning parameters:

##

## minsplit Accuracy Kappa

## 30 0.8320546 0.2098294

## 40 0.8349672 0.2151947

##

## Tuning parameter 'minbucket' was held constant at a value of 25

##

## Tuning parameter 'minprob' was held constant at a value of 0.1

##

## Tuning parameter 'maxheight' was held constant at a value of 4

## Kappa was used to select the optimal model using the largest value.

## The final values used for the model were minsplit = 40, minbucket =

## 25, minprob = 0.1 and maxheight = 4.

chaid.m3$finalModel

##

## Model formula:

## .outcome ~ Age + BusinessTravel + DailyRate + Department + DistanceFromHome +

## Education + EducationField + EnvironmentSatisfaction + Gender +

## HourlyRate + JobInvolvement + JobLevel + JobRole + JobSatisfaction +

## MaritalStatus + MonthlyIncome + MonthlyRate + NumCompaniesWorked +

## OverTime + PercentSalaryHike + PerformanceRating + RelationshipSatisfaction +

## StockOptionLevel + TotalWorkingYears + TrainingTimesLastYear +

## WorkLifeBalance + YearsAtCompany + YearsInCurrentRole + YearsSinceLastPromotion +

## YearsWithCurrManager

##

## Fitted party:

## [1] root

## | [2] OverTime in No

## | | [3] YearsAtCompany in [0,2]

## | | | [4] Age in [18,29], (29,34]

## | | | | [5] StockOptionLevel in 0: No (n = 43, err = 48.8%)

## | | | | [6] StockOptionLevel in 1, 2, 3: No (n = 45, err = 15.6%)

## | | | [7] Age in (34,38], (38,45], (45,60]: No (n = 77, err = 7.8%)

## | | [8] YearsAtCompany in (2,5], (5,7], (7,10], (10,40]

## | | | [9] WorkLifeBalance in Bad: No (n = 36, err = 19.4%)

## | | | [10] WorkLifeBalance in Good, Better, Best

## | | | | [11] Department in Human\_Resources, Sales: No (n = 183, err = 9.8%)

## | | | | [12] Department in Research\_Development: No (n = 347, err = 4.0%)

## | [13] OverTime in Yes

## | | [14] JobLevel in 1

## | | | [15] JobRole in Healthcare\_Representative, Human\_Resources, Laboratory\_Technician, Manager, Manufacturing\_Director, Research\_Director, Sales\_Executive, Sales\_Representative

## | | | | [16] YearsInCurrentRole in [0,1], (2,4], (4,7]: Yes (n = 35, err = 17.1%)

## | | | | [17] YearsInCurrentRole in (1,2], (7,18]: No (n = 29, err = 44.8%)

## | | | [18] JobRole in Research\_Scientist: No (n = 53, err = 35.8%)

## | | [19] JobLevel in 2, 3, 4, 5

## | | | [20] Gender in Female: No (n = 86, err = 9.3%)

## | | | [21] Gender in Male

## | | | | [22] MaritalStatus in Divorced, Married: No (n = 71, err = 18.3%)

## | | | | [23] MaritalStatus in Single: No (n = 25, err = 44.0%)

##

## Number of inner nodes: 11

## Number of terminal nodes: 12

confusionMatrix(chaid.m3)

## Cross-Validated (10 fold) Confusion Matrix

##

## (entries are percentual average cell counts across resamples)

##

## Reference

## Prediction No Yes

## No 80.0 12.6

## Yes 3.9 3.5

##

## Accuracy (average) : 0.835

confusionMatrix(predict(chaid.m3), y)

## Confusion Matrix and Statistics

##

## Reference

## Prediction No Yes

## No 858 137

## Yes 6 29

##

## Accuracy : 0.8612

## 95% CI : (0.8385, 0.8817)

## No Information Rate : 0.8388

## P-Value [Acc > NIR] : 0.02656

##

## Kappa : 0.2463

## Mcnemar's Test P-Value : < 2e-16

##

## Sensitivity : 0.9931

## Specificity : 0.1747

## Pos Pred Value : 0.8623

## Neg Pred Value : 0.8286

## Prevalence : 0.8388

## Detection Rate : 0.8330

## Detection Prevalence : 0.9660

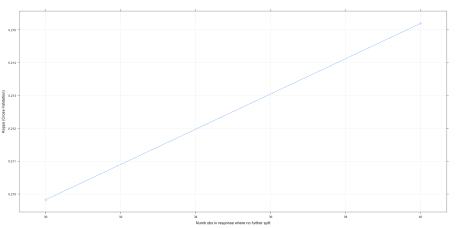
## Balanced Accuracy : 0.5839

##

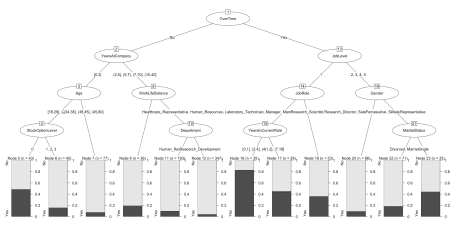
## 'Positive' Class : No

##

plot(chaid.m3)



plot(chaid.m3$finalModel)



A quick reminder that you can get relative variable importance with  
varImp. And of course the all important look at how well we predicted  
against our held out test data set.

varImp(chaid.m3)

## ROC curve variable importance

##

## only 20 most important variables shown (out of 30)

##

## Importance

## OverTime 100.00

## YearsInCurrentRole 90.81

## YearsAtCompany 90.41

## MonthlyIncome 87.08

## JobLevel 84.36

## TotalWorkingYears 80.04

## YearsWithCurrManager 79.78

## StockOptionLevel 69.51

## MaritalStatus 65.96

## Age 59.31

## JobSatisfaction 44.86

## JobInvolvement 44.27

## DistanceFromHome 36.80

## EnvironmentSatisfaction 32.15

## WorkLifeBalance 31.63

## DailyRate 30.23

## JobRole 29.94

## NumCompaniesWorked 28.67

## Department 25.79

## HourlyRate 19.81

confusionMatrix(predict(chaid.m3, newdata = test), test$Attrition)

## Confusion Matrix and Statistics

##

## Reference

## Prediction No Yes

## No 365 67

## Yes 4 4

##

## Accuracy : 0.8386

## 95% CI : (0.8009, 0.8718)

## No Information Rate : 0.8386

## P-Value [Acc > NIR] : 0.5316

##

## Kappa : 0.0709

## Mcnemar's Test P-Value : 1.866e-13

##

## Sensitivity : 0.98916

## Specificity : 0.05634

## Pos Pred Value : 0.84491

## Neg Pred Value : 0.50000

## Prevalence : 0.83864

## Detection Rate : 0.82955

## Detection Prevalence : 0.98182

## Balanced Accuracy : 0.52275

##

## 'Positive' Class : No

##

One last exercise might also be fruitful. Suppose the only thing you  
wanted to tell chaid was how deeply it was allowed to go in the tree.  
Let’s run a simple example where we use all the defaults but force  
either a two level or three level solution.

# set up tuning grid cgpCHAID

search\_grid <- expand.grid(

minsplit = c(30),

minprob = .01,

minbucket = 7,

maxheight = 3:4

)

# train model

chaid.m4 <- train(

x = x,

y = y,

method = cgpCHAID,

metric = "Kappa",

trControl = train\_control,

tuneGrid = search\_grid

)

Those simple steps produce chaid.m4 which we can then investigate in  
the usual way.

chaid.m4

## CGP CHAID

##

## 1030 samples

## 30 predictor

## 2 classes: 'No', 'Yes'

##

## No pre-processing

## Resampling: Cross-Validated (10 fold)

## Summary of sample sizes: 927, 926, 927, 926, 928, 927, ...

## Resampling results across tuning parameters:

##

## maxheight Accuracy Kappa

## 3 0.8417388 0.2306686

## 4 0.8291923 0.1885956

##

## Tuning parameter 'minsplit' was held constant at a value of 30

##

## Tuning parameter 'minbucket' was held constant at a value of 7

##

## Tuning parameter 'minprob' was held constant at a value of 0.01

## Kappa was used to select the optimal model using the largest value.

## The final values used for the model were minsplit = 30, minbucket =

## 7, minprob = 0.01 and maxheight = 3.

chaid.m4$finalModel

##

## Model formula:

## .outcome ~ Age + BusinessTravel + DailyRate + Department + DistanceFromHome +

## Education + EducationField + EnvironmentSatisfaction + Gender +

## HourlyRate + JobInvolvement + JobLevel + JobRole + JobSatisfaction +

## MaritalStatus + MonthlyIncome + MonthlyRate + NumCompaniesWorked +

## OverTime + PercentSalaryHike + PerformanceRating + RelationshipSatisfaction +

## StockOptionLevel + TotalWorkingYears + TrainingTimesLastYear +

## WorkLifeBalance + YearsAtCompany + YearsInCurrentRole + YearsSinceLastPromotion +

## YearsWithCurrManager

##

## Fitted party:

## [1] root

## | [2] OverTime in No

## | | [3] YearsAtCompany in [0,2]

## | | | [4] Age in [18,29], (29,34]: No (n = 88, err = 31.8%)

## | | | [5] Age in (34,38], (38,45], (45,60]: No (n = 77, err = 7.8%)

## | | [6] YearsAtCompany in (2,5], (5,7], (7,10], (10,40]

## | | | [7] WorkLifeBalance in Bad: No (n = 36, err = 19.4%)

## | | | [8] WorkLifeBalance in Good, Better, Best: No (n = 530, err = 6.0%)

## | [9] OverTime in Yes

## | | [10] JobLevel in 1

## | | | [11] JobRole in Healthcare\_Representative, Human\_Resources, Laboratory\_Technician, Manager, Manufacturing\_Director, Research\_Director, Sales\_Executive, Sales\_Representative: Yes (n = 64, err = 34.4%)

## | | | [12] JobRole in Research\_Scientist: No (n = 53, err = 35.8%)

## | | [13] JobLevel in 2, 3, 4, 5

## | | | [14] Gender in Female: No (n = 86, err = 9.3%)

## | | | [15] Gender in Male: No (n = 96, err = 25.0%)

##

## Number of inner nodes: 7

## Number of terminal nodes: 8

confusionMatrix(chaid.m4)

## Cross-Validated (10 fold) Confusion Matrix

##

## (entries are percentual average cell counts across resamples)

##

## Reference

## Prediction No Yes

## No 80.6 12.5

## Yes 3.3 3.6

##

## Accuracy (average) : 0.8417

confusionMatrix(predict(chaid.m4), y)

## Confusion Matrix and Statistics

##

## Reference

## Prediction No Yes

## No 842 124

## Yes 22 42

##

## Accuracy : 0.8583

## 95% CI : (0.8354, 0.879)

## No Information Rate : 0.8388

## P-Value [Acc > NIR] : 0.04743

##

## Kappa : 0.3027

## Mcnemar's Test P-Value : < 2e-16

##

## Sensitivity : 0.9745

## Specificity : 0.2530

## Pos Pred Value : 0.8716

## Neg Pred Value : 0.6563

## Prevalence : 0.8388

## Detection Rate : 0.8175

## Detection Prevalence : 0.9379

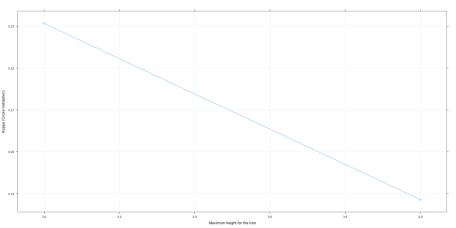
## Balanced Accuracy : 0.6138

##

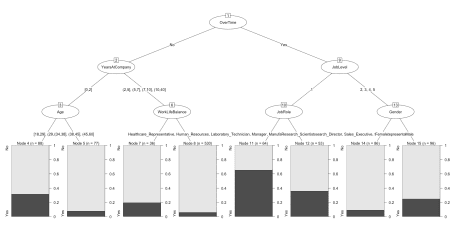
## 'Positive' Class : No

##

plot(chaid.m4)



plot(chaid.m4$finalModel)



Although this post is more about explaining how to use the tools than it  
is about actually fitting this fictional data, let’s review all four of  
the models we built for comparative purposes.

confusionMatrix(predict(chaid.m1, newdata = test), test$Attrition)

## Confusion Matrix and Statistics

##

## Reference

## Prediction No Yes

## No 357 64

## Yes 12 7

##

## Accuracy : 0.8273

## 95% CI : (0.7886, 0.8614)

## No Information Rate : 0.8386

## P-Value [Acc > NIR] : 0.7642

##

## Kappa : 0.0938

## Mcnemar's Test P-Value : 4.913e-09

##

## Sensitivity : 0.96748

## Specificity : 0.09859

## Pos Pred Value : 0.84798

## Neg Pred Value : 0.36842

## Prevalence : 0.83864

## Detection Rate : 0.81136

## Detection Prevalence : 0.95682

## Balanced Accuracy : 0.53304

##

## 'Positive' Class : No

##

confusionMatrix(predict(chaid.m2, newdata = test), test$Attrition)

## Confusion Matrix and Statistics

##

## Reference

## Prediction No Yes

## No 351 50

## Yes 18 21

##

## Accuracy : 0.8455

## 95% CI : (0.8082, 0.8779)

## No Information Rate : 0.8386

## P-Value [Acc > NIR] : 0.3779937

##

## Kappa : 0.3019

## Mcnemar's Test P-Value : 0.0001704

##

## Sensitivity : 0.9512

## Specificity : 0.2958

## Pos Pred Value : 0.8753

## Neg Pred Value : 0.5385

## Prevalence : 0.8386

## Detection Rate : 0.7977

## Detection Prevalence : 0.9114

## Balanced Accuracy : 0.6235

##

## 'Positive' Class : No

##

confusionMatrix(predict(chaid.m3, newdata = test), test$Attrition)

## Confusion Matrix and Statistics

##

## Reference

## Prediction No Yes

## No 365 67

## Yes 4 4

##

## Accuracy : 0.8386

## 95% CI : (0.8009, 0.8718)

## No Information Rate : 0.8386

## P-Value [Acc > NIR] : 0.5316

##

## Kappa : 0.0709

## Mcnemar's Test P-Value : 1.866e-13

##

## Sensitivity : 0.98916

## Specificity : 0.05634

## Pos Pred Value : 0.84491

## Neg Pred Value : 0.50000

## Prevalence : 0.83864

## Detection Rate : 0.82955

## Detection Prevalence : 0.98182

## Balanced Accuracy : 0.52275

##

## 'Positive' Class : No

##

confusionMatrix(predict(chaid.m4, newdata = test), test$Attrition)

## Confusion Matrix and Statistics

##

## Reference

## Prediction No Yes

## No 362 64

## Yes 7 7

##

## Accuracy : 0.8386

## 95% CI : (0.8009, 0.8718)

## No Information Rate : 0.8386

## P-Value [Acc > NIR] : 0.5316

##

## Kappa : 0.1178

## Mcnemar's Test P-Value : 3.012e-11

##

## Sensitivity : 0.98103

## Specificity : 0.09859

## Pos Pred Value : 0.84977

## Neg Pred Value : 0.50000

## Prevalence : 0.83864

## Detection Rate : 0.82273

## Detection Prevalence : 0.96818

## Balanced Accuracy : 0.53981

##

## 'Positive' Class : No

##

At this juncture we’re faced with the same problem we had in my last  
post. We’re drowning in data from the individual confusionMatrix  
results. To do that we need to:

1. Make a named list called modellist that contains our 4 models  
   with a descriptive name for each
2. Use map from purrr to apply the predict command to each model  
   in turn to our test dataset
3. Pipe those results to a second map command to generate a confusion  
   matrix comparing our predictions to test$Attrition which are the  
   actual outcomes.
4. Pipe those results to a complex map\_dfr (that I explained last  
   time) that creates a dataframe of all the results with each CHAID  
   model as a row.
5. Show us the names of the columns we have available.

modellist <- list("Default tune" = chaid.m1,

"a2 & a4 stricter" = chaid.m2,

"Custom parameters" = chaid.m3,

"3 or 4 levels" = chaid.m4)

CHAIDResults <- map(modellist, ~ predict(.x, newdata = test)) %>%

map(~ confusionMatrix(test$Attrition, .x)) %>%

map\_dfr(~ cbind(as.data.frame(t(.x$overall)),as.data.frame(t(.x$byClass))), .id = "ModelNumb")

names(CHAIDResults)

## [1] "ModelNumb" "Accuracy" "Kappa"

## [4] "AccuracyLower" "AccuracyUpper" "AccuracyNull"

## [7] "AccuracyPValue" "McnemarPValue" "Sensitivity"

## [10] "Specificity" "Pos Pred Value" "Neg Pred Value"

## [13] "Precision" "Recall" "F1"

## [16] "Prevalence" "Detection Rate" "Detection Prevalence"

## [19] "Balanced Accuracy"

From the list of available columns let’s use dplyr to select just the  
columns we want, round the numeric columns to 3 digits and then use  
kable to make a pretty table that is much easier to understand.

CHAIDResults %>%

select("ModelNumb", "Accuracy", "Kappa", "Sensitivity", "Specificity", "Neg Pred Value", "F1", "Balanced Accuracy") %>%

mutate\_if(is.numeric,funs(round(.,3))) %>%

kable("html") %>%

kable\_styling(bootstrap\_options = c("striped", "hover", "condensed", "responsive"))

| **ModelNumb** | **Accuracy** | **Kappa** | **Sensitivity** | **Specificity** | **Neg Pred Value** | **F1** | **Balanced Accuracy** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Default tune | 0.827 | 0.094 | 0.848 | 0.368 | 0.099 | 0.904 | 0.608 |
| a2 & a4 stricter | 0.845 | 0.302 | 0.875 | 0.538 | 0.296 | 0.912 | 0.707 |
| Custom parameters | 0.839 | 0.071 | 0.845 | 0.500 | 0.056 | 0.911 | 0.672 |
| 3 or 4 levels | 0.839 | 0.118 | 0.850 | 0.500 | 0.099 | 0.911 | 0.675 |

By nearly every measure we care about, chaid.m2 (where the best fit was  
alpha2 = 0.05 and alpha4 = 0.001) clearly emerges as the best predictor  
against out test dataset. **N.B.** notice that if you only focus on  
the default accuracy measure, the models are all very close. But if you  
focus on more precise measures like Kappa and Negative Predictive Value  
(which in this case is a great indicator of how well we are specifically  
getting our prediction of attrition correct – compared to the more  
common case of predicting that people will stay)

It’s a very simple and parsimonious model, where we only need to know  
three things about the staff member to get pretty accurate predictions;  
Overtime, YearsAtCompany, and JobLevel. It’s very clear that some  
of the other variables may be at work here but we should acquire more  
data to make that assessment rather than trying to overpredict with the  
data we have on hand.