CaseStudy2DDS

Rinku Lichti

11/25/2020

# Install and Load Libraries as needed  
library(pacman)  
p\_load("summarytools", "dplyr", "ggplot2", "ggsci", "caret", "corrplot", "GGally", "pROC", "readr",  
 "randomForestExplainer", "cowplot")

# Load the data

# Summarize the raw data

#print(dfSummary(df, graph.magnif = 0.75), method = 'browser')

From dfSummary, I can see: \* Age is normally distributed \* These are right skewed: DistanceFromHome, MonthlyIncome, PercentSalaryHike, TotalWorkingYears, YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion, and YearsWithCurrManager. \* These have static values: Over18, StandardHours, EmployeeCount \* ID is likely not helpful for prediction, but may be helpful for investigation

clean.for.eda <- function(dirty) {  
 dfTemp <- dirty  
   
 dfTemp$age\_group <- cut(dfTemp$Age,   
 breaks = c(17,22,26,30,34,38,42,46,50,Inf),   
 labels = c("18-22", "22-26", "26-30", "30-34", "34-38", "38-42", "42-46", "46-50", "50+"))  
  
 dfTemp <- dfTemp %>% relocate(Attrition, .after = last\_col())  
  
 dfTemp  
}  
  
df <- clean.for.eda(df)

str(df)

## 'data.frame': 870 obs. of 37 variables:  
## $ ID : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ Age : int 32 40 35 32 24 27 41 37 34 34 ...  
## $ BusinessTravel : Factor w/ 3 levels "Non-Travel","Travel\_Frequently",..: 3 3 2 3 2 2 3 3 3 2 ...  
## $ DailyRate : int 117 1308 200 801 567 294 1283 309 1333 653 ...  
## $ Department : Factor w/ 3 levels "Human Resources",..: 3 2 2 3 2 2 2 3 3 2 ...  
## $ DistanceFromHome : int 13 14 18 1 2 10 5 10 10 10 ...  
## $ Education : int 4 3 2 4 1 2 5 4 4 4 ...  
## $ EducationField : Factor w/ 6 levels "Human Resources",..: 2 4 2 3 6 2 4 2 2 6 ...  
## $ EmployeeCount : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ EmployeeNumber : int 859 1128 1412 2016 1646 733 1448 1105 1055 1597 ...  
## $ EnvironmentSatisfaction : int 2 3 3 3 1 4 2 4 3 4 ...  
## $ Gender : Factor w/ 2 levels "Female","Male": 2 2 2 1 1 2 2 1 1 2 ...  
## $ HourlyRate : int 73 44 60 48 32 32 90 88 87 92 ...  
## $ JobInvolvement : int 3 2 3 3 3 3 4 2 3 2 ...  
## $ JobLevel : int 2 5 3 3 1 3 1 2 1 2 ...  
## $ JobRole : Factor w/ 9 levels "Healthcare Representative",..: 8 6 5 8 7 5 7 8 9 1 ...  
## $ JobSatisfaction : int 4 3 4 4 4 1 3 4 3 3 ...  
## $ MaritalStatus : Factor w/ 3 levels "Divorced","Married",..: 1 3 3 2 3 1 2 1 2 2 ...  
## $ MonthlyIncome : int 4403 19626 9362 10422 3760 8793 2127 6694 2220 5063 ...  
## $ MonthlyRate : int 9250 17544 19944 24032 17218 4809 5561 24223 18410 15332 ...  
## $ NumCompaniesWorked : int 2 1 2 1 1 1 2 2 1 1 ...  
## $ Over18 : Factor w/ 1 level "Y": 1 1 1 1 1 1 1 1 1 1 ...  
## $ OverTime : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 2 2 2 1 ...  
## $ PercentSalaryHike : int 11 14 11 19 13 21 12 14 19 14 ...  
## $ PerformanceRating : int 3 3 3 3 3 4 3 3 3 3 ...  
## $ RelationshipSatisfaction: int 3 1 3 3 3 3 1 3 4 2 ...  
## $ StandardHours : int 80 80 80 80 80 80 80 80 80 80 ...  
## $ StockOptionLevel : int 1 0 0 2 0 2 0 3 1 1 ...  
## $ TotalWorkingYears : int 8 21 10 14 6 9 7 8 1 8 ...  
## $ TrainingTimesLastYear : int 3 2 2 3 2 4 5 5 2 3 ...  
## $ WorkLifeBalance : int 2 4 3 3 3 2 2 3 3 2 ...  
## $ YearsAtCompany : int 5 20 2 14 6 9 4 1 1 8 ...  
## $ YearsInCurrentRole : int 2 7 2 10 3 7 2 0 1 2 ...  
## $ YearsSinceLastPromotion : int 0 4 2 5 1 1 0 0 0 7 ...  
## $ YearsWithCurrManager : int 3 9 2 7 3 7 3 0 0 7 ...  
## $ age\_group : Factor w/ 9 levels "18-22","22-26",..: 4 6 5 4 2 3 6 5 4 4 ...  
## $ Attrition : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...

# Visualize Attrition

## Attrition Distribution

table(df$Attrition)

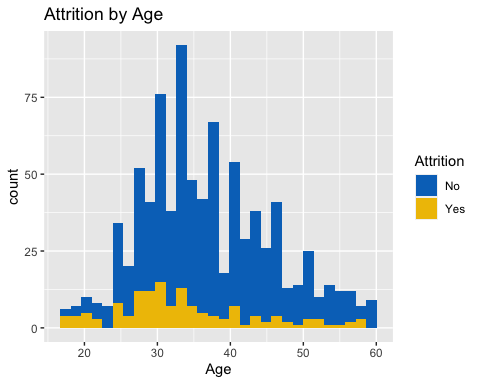
##   
## No Yes   
## 730 140

Attrition records represent a small minority of the people represented in this data.

## Attrition Bar Plot By Age

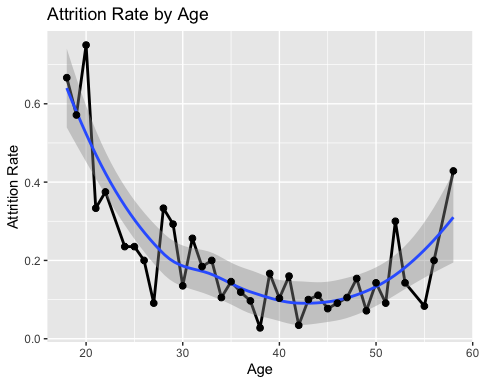
ggplot(df) +   
 geom\_histogram(mapping = aes(x=Age, fill=Attrition)) +   
 ggtitle("Distribution of Attrition by Age") +   
 scale\_fill\_jco() +  
 ggtitle("Attrition by Age")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

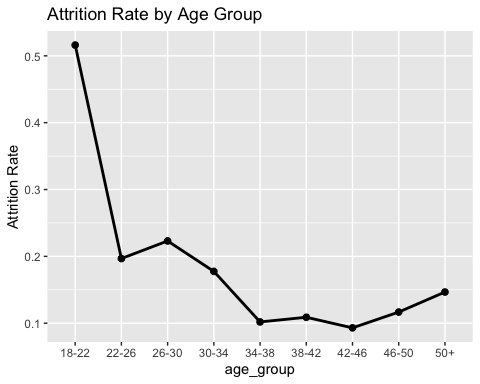


# Helper function to plot attrition rate by anything...  
plot\_attrition\_rate\_by <- function(data, column) {  
 dfTemp <- data %>%  
 group\_by\_(column) %>%  
 count(Attrition) %>%  
 mutate(AttritionRate = n/sum(n)) %>%  
 filter(Attrition == "Yes")  
  
 ggplot(dfTemp, aes\_string(x=column, y="AttritionRate", group=1)) +  
 geom\_line(size=1) +  
 geom\_smooth() +   
 geom\_point(size=2) +   
 scale\_color\_jco() +  
 scale\_fill\_jco() +  
 ggtitle(paste("Attrition Rate by", column)) +   
 ylab("Attrition Rate")  
}

trend1 <- plot\_attrition\_rate\_by(df, "Age")  
trend1

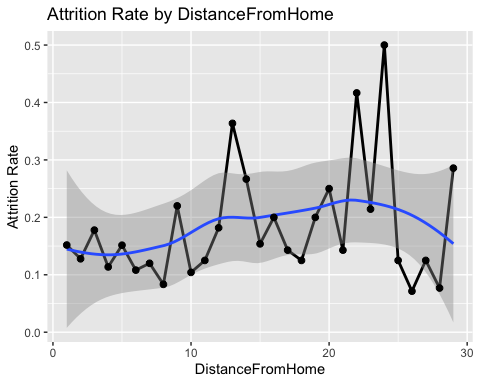


df.age\_eda <- df %>%  
 group\_by(age\_group) %>%  
 count(Attrition) %>%  
 mutate(age\_group\_attr = n/sum(n)) %>%  
 filter(Attrition == "Yes")  
  
ggplot(df.age\_eda, aes(x=age\_group, y=age\_group\_attr, group=1)) +  
 geom\_line(size=1) +  
 geom\_point(size=2) +   
 scale\_color\_jco() +  
 scale\_fill\_jco() +  
 ggtitle("Attrition Rate by Age Group") +   
 ylab("Attrition Rate")

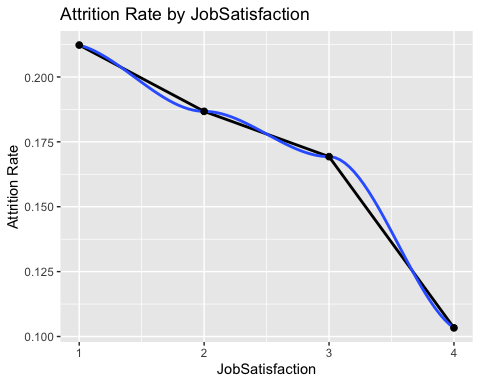


## DistanceFromHome

plot\_attrition\_rate\_by(df, "DistanceFromHome")

 ## JobSatisfaction

trend2 <- plot\_attrition\_rate\_by(df, "JobSatisfaction")  
trend2

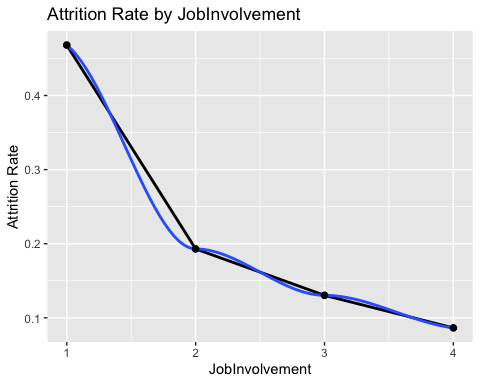


## JobLevel

trend3 <- plot\_attrition\_rate\_by(df, "JobLevel")  
trend3

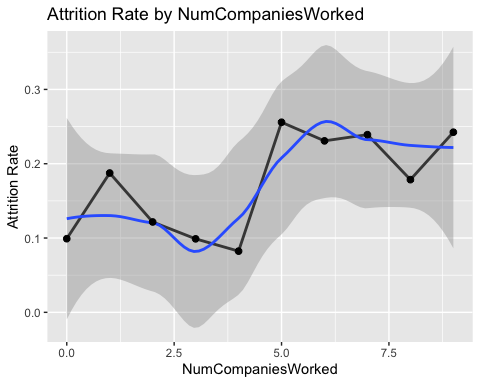
 ## JobInvolvement

trend4 <- plot\_attrition\_rate\_by(df, "JobInvolvement")  
trend4



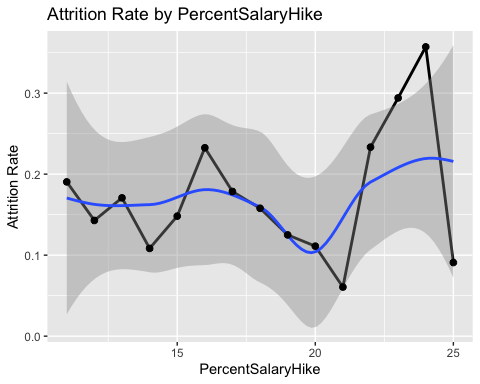
## NumCompaniesWorked

trend5 <- plot\_attrition\_rate\_by(df, "NumCompaniesWorked")  
trend5

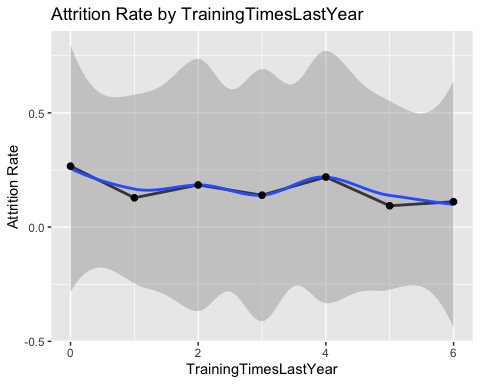


## PercentSalaryHike

trend6 <- plot\_attrition\_rate\_by(df, "PercentSalaryHike")  
trend6

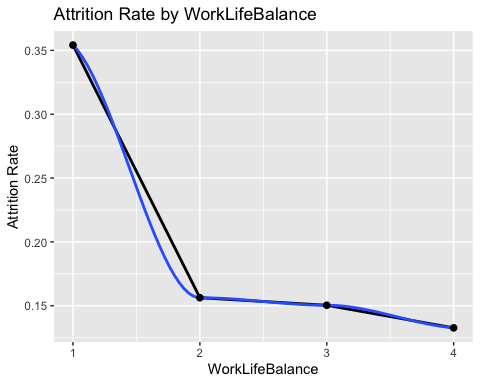
 ## TrainingTimesLastYear

trend7 <- plot\_attrition\_rate\_by(df, "TrainingTimesLastYear")  
trend7



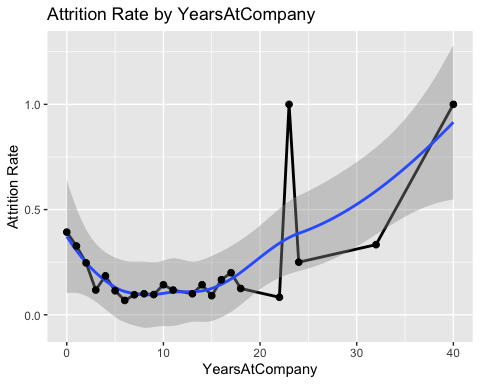
## WorkLifeBalance

trend8 <- plot\_attrition\_rate\_by(df, "WorkLifeBalance")  
trend8

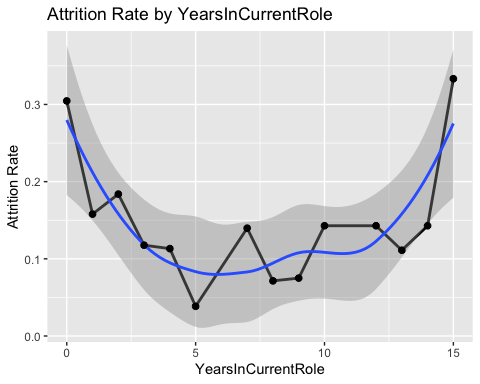


## YearsAtCompany

trend9 <- plot\_attrition\_rate\_by(df, "YearsAtCompany")  
trend9

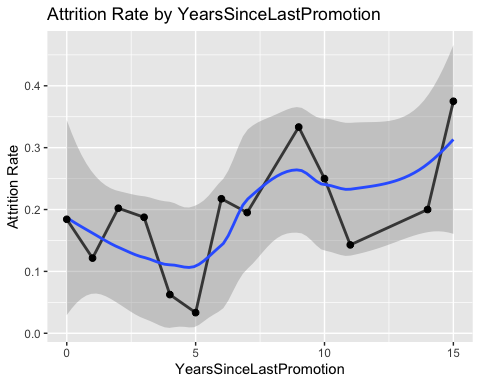
 ## YearsInCurrentRole

trend10 <- plot\_attrition\_rate\_by(df, "YearsInCurrentRole")  
trend10

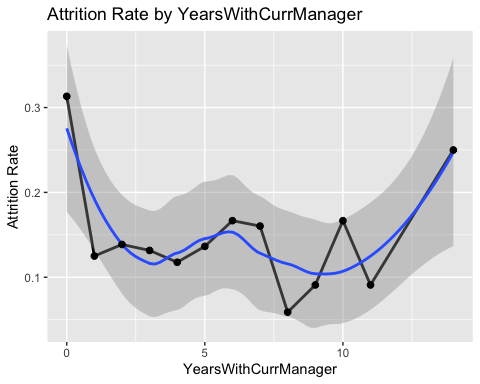


## YearsSinceLastPromotion

trend11 <- plot\_attrition\_rate\_by(df, "YearsSinceLastPromotion")  
trend11

 ## YearsWithCurrManager

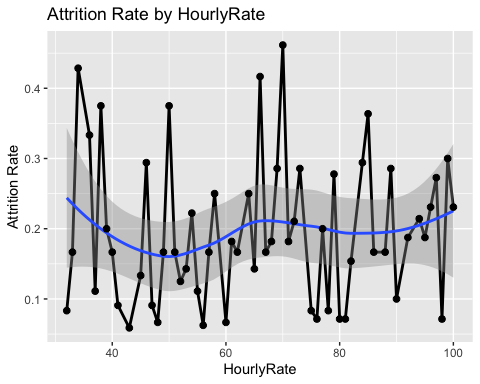
trend12 <- plot\_attrition\_rate\_by(df, "YearsWithCurrManager")  
trend12



## Compensation

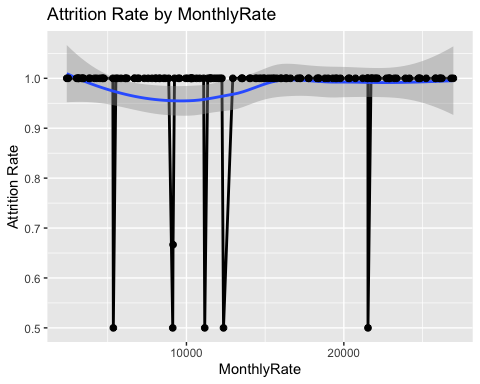
plot\_attrition\_rate\_by(df, "HourlyRate")

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



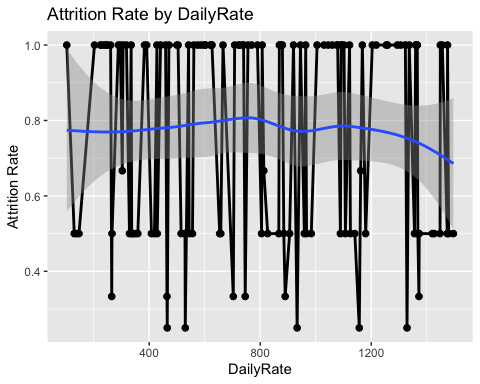
plot\_attrition\_rate\_by(df, "MonthlyRate")

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



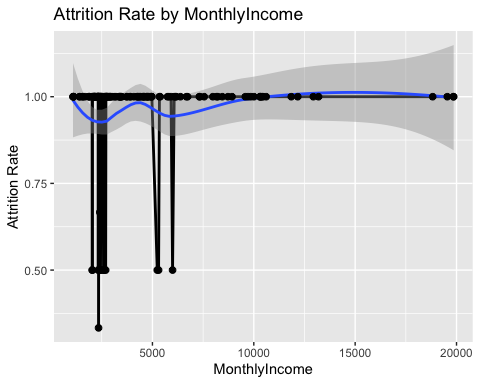
plot\_attrition\_rate\_by(df, "DailyRate")

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



plot\_attrition\_rate\_by(df, "MonthlyIncome")

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



## Trends Grid

trends <- plot\_grid(trend1, trend2, trend3, trend4, trend5, trend6, trend7, trend8,   
 trend9, trend10, trend11, trend12, ncol = 3, labels = "auto")

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : span too small. fewer data values than degrees of freedom.

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : pseudoinverse used at 0.985

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : neighborhood radius 2.015

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : There are other near singularities as well. 4.0602

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : span too small. fewer  
## data values than degrees of freedom.

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at  
## 0.985

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius  
## 2.015

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition  
## number 0

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : There are other near  
## singularities as well. 4.0602

## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning -  
## Inf

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : span too small. fewer data values than degrees of freedom.

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : pseudoinverse used at 0.98

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : neighborhood radius 2.02

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : There are other near singularities as well. 4.0804

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : span too small. fewer  
## data values than degrees of freedom.

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at  
## 0.98

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius 2.02

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition  
## number 0

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : There are other near  
## singularities as well. 4.0804

## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning -  
## Inf

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : span too small. fewer data values than degrees of freedom.

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : pseudoinverse used at 0.985

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : neighborhood radius 2.015

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : There are other near singularities as well. 4.0602

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : span too small. fewer  
## data values than degrees of freedom.

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at  
## 0.985

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius  
## 2.015

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition  
## number 0

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : There are other near  
## singularities as well. 4.0602

## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning -  
## Inf

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : span too small. fewer data values than degrees of freedom.

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : pseudoinverse used at 0.985

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : neighborhood radius 2.015

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : There are other near singularities as well. 4.0602

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : span too small. fewer  
## data values than degrees of freedom.

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at  
## 0.985

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius  
## 2.015

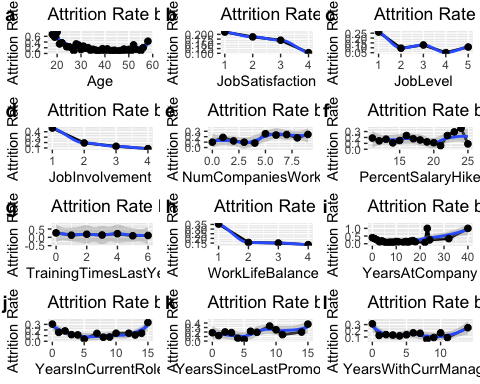
## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition  
## number 0

## Warning in predLoess(object$y, object$x, newx = if  
## (is.null(newdata)) object$x else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : There are other near  
## singularities as well. 4.0602

## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning -  
## Inf

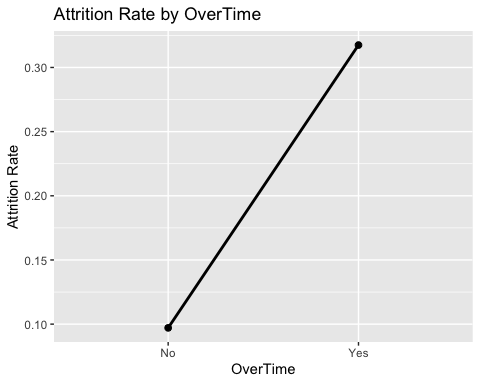
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

trends



## Overtime

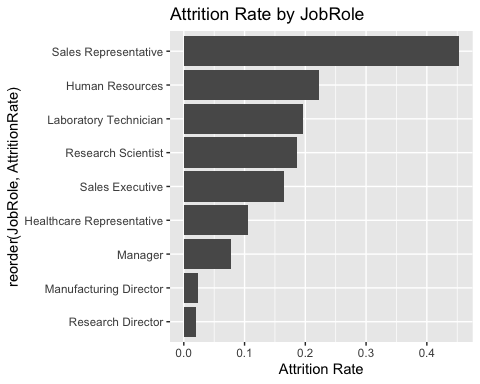
plot\_attrition\_rate\_by(df, "OverTime")



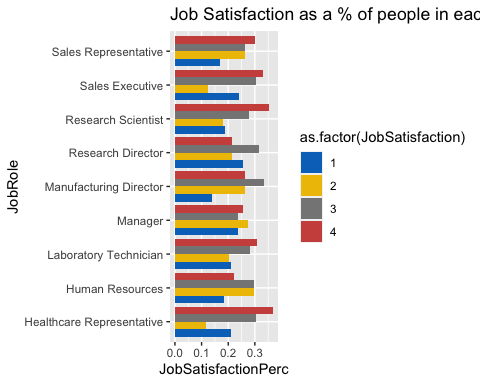
Attrition is significantly higher for those with OverTime

## JobRole

plot\_attrition\_rate\_hbar\_by <- function(data, column) {  
 dfTemp <- data %>%  
 group\_by\_(column) %>%  
 count(Attrition) %>%  
 mutate(AttritionRate = n/sum(n)) %>%  
 filter(Attrition == "Yes")  
 dfTemp <- dfTemp %>%  
 arrange(desc(AttritionRate))  
  
 ggplot(dfTemp, aes\_string(x=paste0("reorder(",column,", AttritionRate)"), y="AttritionRate")) +  
 geom\_col() +  
 coord\_flip() +   
 scale\_color\_jco() +  
 scale\_fill\_jco() +  
 ggtitle(paste("Attrition Rate by", column)) +   
 ylab("Attrition Rate")  
}  
  
  
plot\_attrition\_rate\_hbar\_by(df, "JobRole")

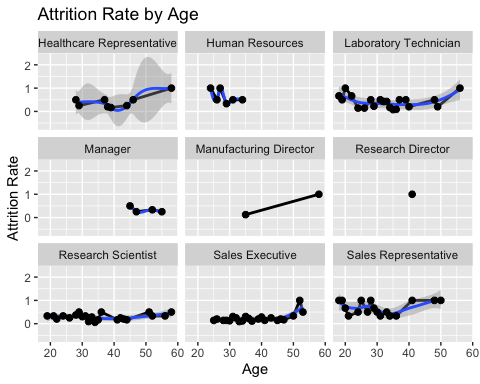


dfTemp <- df  
dfTemp$JobSatisfactionF = as.factor(df$JobSatisfaction)  
  
dfTemp <- dfTemp %>%  
 group\_by(JobRole) %>%  
 count(JobSatisfaction) %>%  
 mutate(JobSatisfactionPerc = n/sum(n))  
  
ggplot(dfTemp, aes(fill=as.factor(JobSatisfaction), y=JobSatisfactionPerc, x=JobRole)) +   
 geom\_bar(position="dodge", stat="identity") +   
 coord\_flip() +   
 scale\_color\_jco() +  
 scale\_fill\_jco() +  
 ggtitle("Job Satisfaction as a % of people in each Job Role")



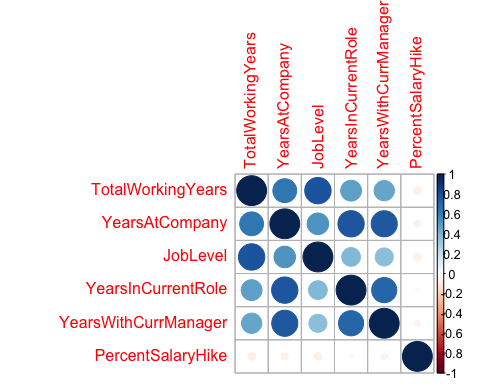
## JobRole by Age

plot\_attrition\_rate\_and\_facet\_by <- function(data, column, facet\_column) {  
 dfTemp <- data %>%  
 group\_by\_(column, facet\_column) %>%  
 count(Attrition) %>%  
 mutate(AttritionRate = n/sum(n)) %>%  
 filter(Attrition == "Yes")  
  
 ggplot(dfTemp, aes\_string(x=column, y="AttritionRate", group=1)) +  
 geom\_line(size=1) +  
 geom\_smooth() +   
 geom\_point(size=2) +   
 scale\_color\_jco() +  
 scale\_fill\_jco() +  
 facet\_wrap(facet\_column) +   
 ggtitle(paste("Attrition Rate by", column)) +   
 ylab("Attrition Rate")  
}  
  
plot\_attrition\_rate\_and\_facet\_by(df, "Age", "JobRole")



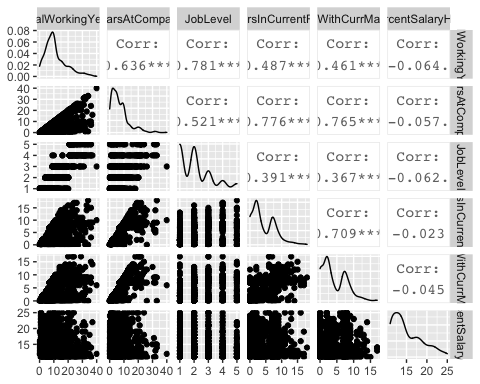
# Checking for correlation

# Look for high correlation  
  
# Only look at numeric fields, and omit ones with constant values  
dfTemp <- df %>% dplyr::select(c(-StandardHours, -Over18, -EmployeeCount, -ID))  
df.numeric <- dfTemp[, sapply(dfTemp, is.numeric)]  
df.without\_na <- na.omit(df.numeric)  
  
corrs = cor(df.without\_na) # Calculate correlations between all variables  
high\_corrs = findCorrelation(corrs, cutoff=0.5)  
corrs = cor(df.without\_na[,high\_corrs]) # get a data frame with only highly correlated variables  
  
#Create corrplot for numeric variables  
corrplot(corrs)



While there are some intuitive correlations, such as YearsAtCompany and YearsWithCurrentManager, I don’t see why we should exclude any of these, which may contribute unique value.

ggpairs(df.without\_na[,high\_corrs])



# Identify the top factors contributing to Monthly Income

set.seed(1234)  
RFcontrol <- rfeControl(  
 functions=rfFuncs,   
 method="cv",   
 number=5,   
 verbose = FALSE)  
  
sizes <- c(1:5, 10, 15, 20)  
  
dfTemp <- df %>% relocate(MonthlyIncome, .after = last\_col())  
  
RFresults <- rfe(dfTemp[,1:(ncol(dfTemp)-1)],   
 dfTemp[[ncol(dfTemp)]],   
 sizes=sizes,   
 rfeControl=RFcontrol)  
  
RFresults

##   
## Recursive feature selection  
##   
## Outer resampling method: Cross-Validated (5 fold)   
##   
## Resampling performance over subset size:  
##   
## Variables RMSE Rsquared MAE RMSESD RsquaredSD MAESD Selected  
## 1 1268.7 0.9240 931.2 108.67 0.014746 92.53   
## 2 1049.1 0.9476 815.4 93.62 0.012998 61.76   
## 3 964.5 0.9558 728.6 82.97 0.011401 40.60 \*  
## 4 1002.8 0.9528 760.8 61.71 0.009838 32.54   
## 5 1065.4 0.9483 814.7 62.87 0.010085 42.58   
## 10 1026.3 0.9504 773.9 67.67 0.011051 37.42   
## 15 1007.4 0.9525 765.5 55.80 0.009236 35.96   
## 20 1030.5 0.9510 788.7 48.53 0.008762 24.81   
## 36 1024.3 0.9514 782.8 55.34 0.009037 33.54   
##   
## The top 3 variables (out of 3):  
## JobLevel, JobRole, TotalWorkingYears

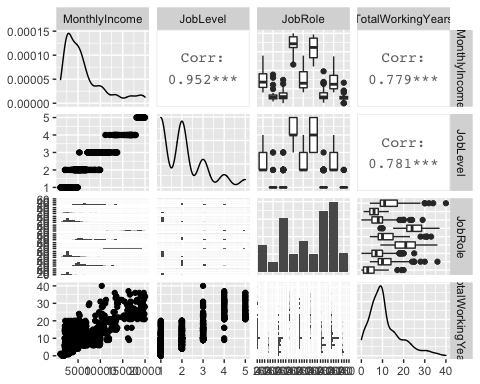
varImp(RFresults)

## Overall  
## JobLevel 37.49832  
## JobRole 28.58650  
## TotalWorkingYears 17.97232

For MonthlyIncome, the top factors are JobLevel, JobRole, and TotalWorkingYears

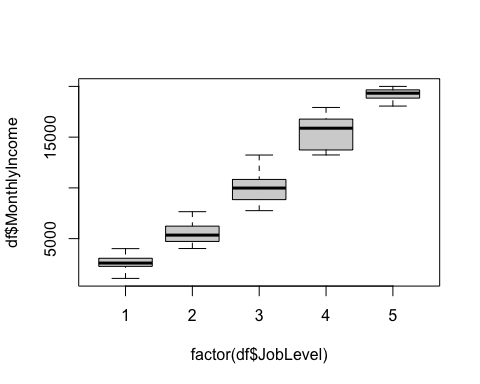
dfTemp <- df %>% select(MonthlyIncome, JobLevel, JobRole, TotalWorkingYears)  
ggpairs(dfTemp)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



## Looking more at JobLevel and MonthlyIncome

qqplot(factor(df$JobLevel), df$MonthlyIncome)



ggplot(df, mapping = aes(x = MonthlyIncome, color = factor(JobLevel), fill = factor(JobLevel))) +   
 geom\_histogram(aes(x = MonthlyIncome)) +  
 facet\_wrap(factor(df$JobLevel)) +   
 scale\_color\_jco() +  
 scale\_fill\_jco()

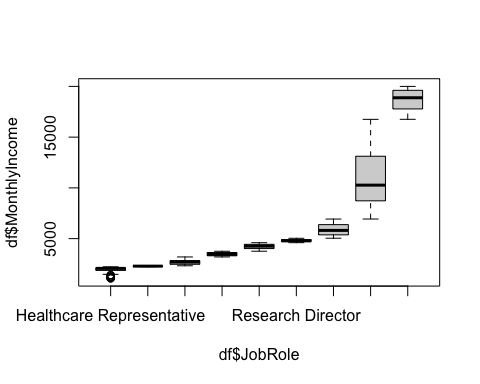
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



It looks like there is a linear relationship between JobLevel and MonthlyIncome, and the respective MonthlyIncome ranges are somewhat normal.

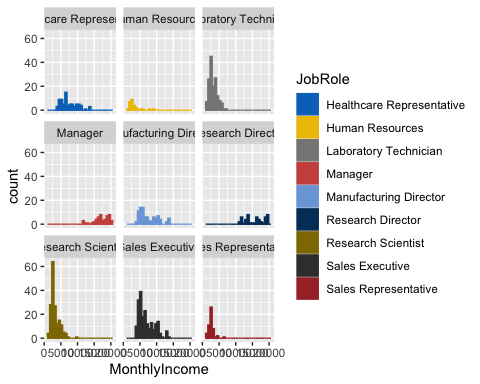
## Looking more at JobRole and MonthlyIncome

qqplot(df$JobRole, df$MonthlyIncome)



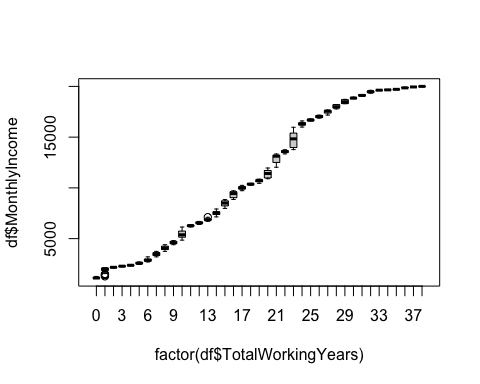
ggplot(df, mapping = aes(x = MonthlyIncome, color = JobRole, fill = JobRole)) +   
 geom\_histogram(aes(x = MonthlyIncome)) +  
 facet\_wrap(df$JobRole) +   
 scale\_color\_jco() +  
 scale\_fill\_jco()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

 It looks like there is a relationship, though non-linear. Also the roles to be are: Research Director and Manager. I’ll run with Research Director.

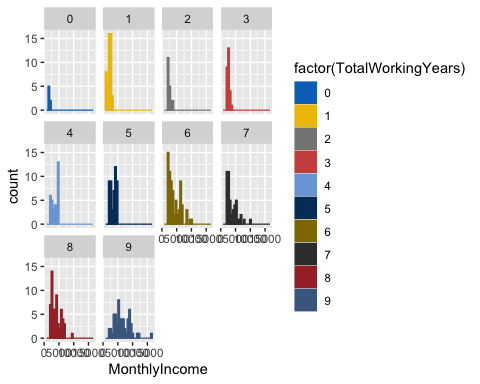
## Looking more at TotalWorkingYears and MonthlyIncome

qqplot(factor(df$TotalWorkingYears), df$MonthlyIncome)



dfTemp <- df %>% filter(TotalWorkingYears < 10)  
ggplot(dfTemp, mapping = aes(x = MonthlyIncome, color = factor(TotalWorkingYears), fill = factor(TotalWorkingYears))) +   
 geom\_histogram(aes(x = MonthlyIncome)) +  
 facet\_wrap(factor(dfTemp$TotalWorkingYears)) +   
 scale\_color\_jco() +  
 scale\_fill\_jco()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

 It looks like TotalWorkingYears has a linear relationship with MonthlyIncome.

# Near Zero Variance

nzv <- nearZeroVar(df, saveMetrics = T)  
nrow(nzv %>% filter(nzv == TRUE))

## [1] 3

No variables are near zero variance

# Identify the top factors contributing to Attrition

set.seed(1234)  
RFcontrol <- rfeControl(  
 functions=rfFuncs,   
 method="cv",   
 number=5,   
 verbose = FALSE)  
  
sizes <- c(1:5, 10, 15, 20)  
  
RFresults <- rfe(df[,1:(ncol(df)-1)],   
 df[[ncol(df)]],   
 sizes=sizes,   
 rfeControl=RFcontrol)  
  
RFresults

##   
## Recursive feature selection  
##   
## Outer resampling method: Cross-Validated (5 fold)   
##   
## Resampling performance over subset size:  
##   
## Variables Accuracy Kappa AccuracySD KappaSD Selected  
## 1 0.8391 0.0000 0.000000 0.00000   
## 2 0.8563 0.2291 0.021116 0.15804   
## 3 0.8483 0.1534 0.014425 0.09534   
## 4 0.8632 0.3392 0.013721 0.04504 \*  
## 5 0.8483 0.2878 0.013231 0.07277   
## 10 0.8506 0.2886 0.008128 0.03522   
## 15 0.8540 0.2603 0.008716 0.04048   
## 20 0.8563 0.2434 0.017241 0.08346   
## 36 0.8563 0.2098 0.014652 0.08594   
##   
## The top 4 variables (out of 4):  
## OverTime, MonthlyIncome, StockOptionLevel, JobRole

varImp(RFresults)

## Overall  
## OverTime 12.533080  
## JobRole 9.021616  
## JobInvolvement 7.215958  
## MonthlyIncome 7.214574  
## StockOptionLevel 6.929491  
## JobLevel 6.144694

dfTemp <- df %>% select(Attrition, JobRole, JobInvolvement, MonthlyIncome, StockOptionLevel, JobLevel)  
x <- ggpairs(dfTemp)

# Clean/Train/Test Split

# - Load and Clean for MonthlyIncome  
  
clean.for.modeling.monthly\_income <- function(dirty) {  
 dfTemp <- dirty %>% dplyr::select(c(-StandardHours, -Over18, -EmployeeCount, -Attrition))  
 dfTemp  
}  
  
load.and.clean.for.monthly\_income <- function(filename, removeID = TRUE) {  
 df <- read.csv(filename, stringsAsFactors=TRUE)  
 df <- clean.for.modeling.monthly\_income(df)  
   
 if (removeID) {  
 df <- df %>% dplyr::select(c(-ID))  
 }  
 if ("MonthlyIncome" %in% names(df)) {  
 df <- df %>% relocate(MonthlyIncome, .after = last\_col())  
 }  
 df  
}  
  
df <- load.and.clean.for.monthly\_income("doc/CaseStudy2-data.csv", TRUE)  
  
set.seed(1234)  
num\_rows <- nrow(df)  
train\_idx <- sample(1:num\_rows, 0.8 \* num\_rows)  
test\_idx <- setdiff(1:num\_rows, train\_idx)  
mi.train <- df[train\_idx, ]  
mi.test <- df[test\_idx, ]  
  
# - Load and Clean for Attrition  
  
clean.for.modeling.attrition <- function(dirty) {  
 dfTemp <- dirty %>% dplyr::select(c(-StandardHours, -Over18, -EmployeeCount))  
  
 dfTemp$age\_group <- cut(dfTemp$Age,   
 breaks = c(17,22,26,30,34,38,42,46,50,Inf),   
 labels = c("18-22", "22-26", "26-30", "30-34", "34-38", "38-42", "42-46", "46-50", "50+"))  
  
 dfTemp  
}  
  
load.and.clean.for.attrition <- function(filename, removeID = TRUE) {  
 df <- read.csv(filename, stringsAsFactors=TRUE)  
 df <- clean.for.modeling.attrition(df)  
   
 if (removeID) {  
 df <- df %>% dplyr::select(c(-ID))  
 }  
 if ("Attrition" %in% names(df)) {  
 df <- df %>% relocate(Attrition, .after = last\_col())  
 }  
 df  
}  
  
df <- load.and.clean.for.attrition("doc/CaseStudy2-data.csv", TRUE)  
  
set.seed(1234)  
num\_rows <- nrow(df)  
train\_idx <- sample(1:num\_rows, 0.8 \* num\_rows)  
test\_idx <- setdiff(1:num\_rows, train\_idx)  
at.train <- df[train\_idx, ]  
at.test <- df[test\_idx, ]

# MonthlyIncome Models

## Linear Regression

lr\_control <- trainControl(method = "cv", num = 5)  
  
mi.fitLR <- train(MonthlyIncome ~ .,   
 data = mi.train,   
 method = "glm",   
 trControl = lr\_control  
 )  
mi.fitLR

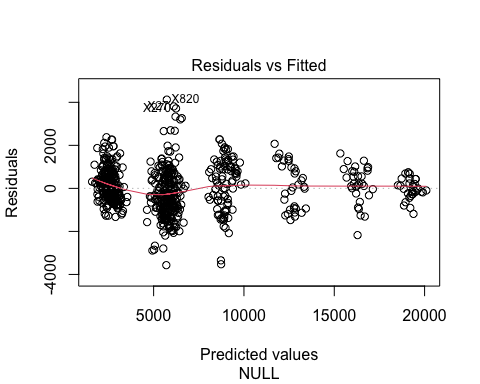
## Generalized Linear Model   
##   
## 696 samples  
## 30 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 556, 558, 557, 557, 556   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 1049.707 0.948227 805.9871

summary(mi.fitLR)

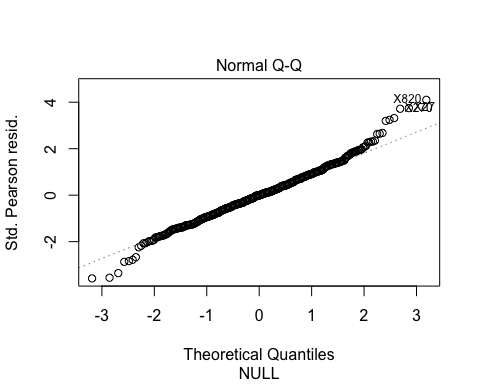
##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3568.9 -620.2 -3.8 605.9 4124.1   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.961e+02 8.346e+02 -0.235 0.814339   
## Age -1.469e-01 6.143e+00 -0.024 0.980925   
## BusinessTravelTravel\_Frequently 1.943e+02 1.544e+02 1.258 0.208683   
## BusinessTravelTravel\_Rarely 4.340e+02 1.300e+02 3.338 0.000891 \*\*\*  
## DailyRate 2.161e-01 9.996e-02 2.162 0.030986 \*   
## `DepartmentResearch & Development` 1.711e+02 4.816e+02 0.355 0.722505   
## DepartmentSales -3.573e+02 5.007e+02 -0.714 0.475692   
## DistanceFromHome -3.118e+00 4.921e+00 -0.634 0.526612   
## Education -2.288e+01 4.056e+01 -0.564 0.572932   
## `EducationFieldLife Sciences` 1.608e+01 3.868e+02 0.042 0.966856   
## EducationFieldMarketing 5.380e+01 4.128e+02 0.130 0.896345   
## EducationFieldMedical -6.193e+01 3.885e+02 -0.159 0.873399   
## EducationFieldOther 3.154e+01 4.209e+02 0.075 0.940297   
## `EducationFieldTechnical Degree` 4.039e+00 4.073e+02 0.010 0.992091   
## EmployeeNumber 7.402e-02 6.683e-02 1.108 0.268422   
## EnvironmentSatisfaction -8.991e+00 3.721e+01 -0.242 0.809148   
## GenderMale 1.027e+02 8.175e+01 1.256 0.209663   
## HourlyRate -3.687e-01 2.030e+00 -0.182 0.855957   
## JobInvolvement 4.149e+01 5.780e+01 0.718 0.473115   
## JobLevel 2.846e+03 9.131e+01 31.163 < 2e-16 \*\*\*  
## `JobRoleHuman Resources` -9.916e+01 5.261e+02 -0.188 0.850560   
## `JobRoleLaboratory Technician` -5.782e+02 1.848e+02 -3.129 0.001830 \*\*   
## JobRoleManager 4.320e+03 3.082e+02 14.018 < 2e-16 \*\*\*  
## `JobRoleManufacturing Director` 2.530e+02 1.858e+02 1.361 0.173862   
## `JobRoleResearch Director` 4.104e+03 2.478e+02 16.564 < 2e-16 \*\*\*  
## `JobRoleResearch Scientist` -2.619e+02 1.855e+02 -1.412 0.158350   
## `JobRoleSales Executive` 3.964e+02 3.874e+02 1.023 0.306589   
## `JobRoleSales Representative` 1.475e+02 4.211e+02 0.350 0.726243   
## JobSatisfaction 4.829e+01 3.637e+01 1.328 0.184745   
## MaritalStatusMarried 2.818e+01 1.098e+02 0.257 0.797581   
## MaritalStatusSingle -8.443e+01 1.492e+02 -0.566 0.571612   
## MonthlyRate -7.964e-03 5.803e-03 -1.372 0.170397   
## NumCompaniesWorked 8.305e-01 1.849e+01 0.045 0.964180   
## OverTimeYes -2.931e+01 8.911e+01 -0.329 0.742337   
## PercentSalaryHike 3.567e+01 1.719e+01 2.076 0.038324 \*   
## PerformanceRating -3.643e+02 1.753e+02 -2.078 0.038108 \*   
## RelationshipSatisfaction -5.236e-01 3.612e+01 -0.014 0.988439   
## StockOptionLevel -8.392e+01 6.463e+01 -1.299 0.194548   
## TotalWorkingYears 4.233e+01 1.191e+01 3.554 0.000407 \*\*\*  
## TrainingTimesLastYear 1.726e+01 3.228e+01 0.535 0.592988   
## WorkLifeBalance -4.413e+01 5.795e+01 -0.761 0.446665   
## YearsAtCompany -1.098e+01 1.531e+01 -0.717 0.473437   
## YearsInCurrentRole 1.090e+01 1.939e+01 0.562 0.574198   
## YearsSinceLastPromotion 4.403e+01 1.682e+01 2.617 0.009067 \*\*   
## YearsWithCurrManager -2.690e+01 1.913e+01 -1.406 0.160137   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 1070298)  
##   
## Null deviance: 1.4935e+10 on 695 degrees of freedom  
## Residual deviance: 6.9676e+08 on 651 degrees of freedom  
## AIC: 11684  
##   
## Number of Fisher Scoring iterations: 2

It’s sad that Age is negatively correlated with MonthlyIncome. But not as negatively as working in Human Resources. Or being Single. Ouch.

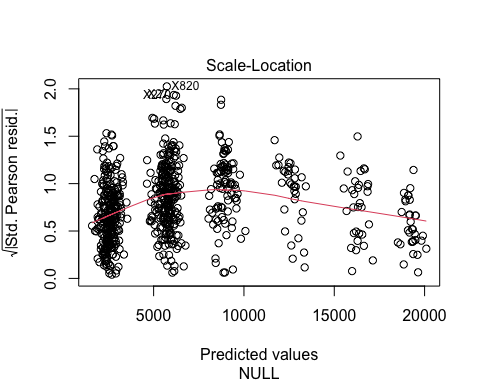
plot(mi.fitLR$finalModel, 1)



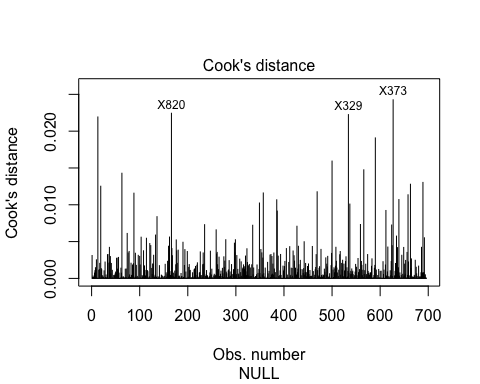
plot(mi.fitLR$finalModel, 2)



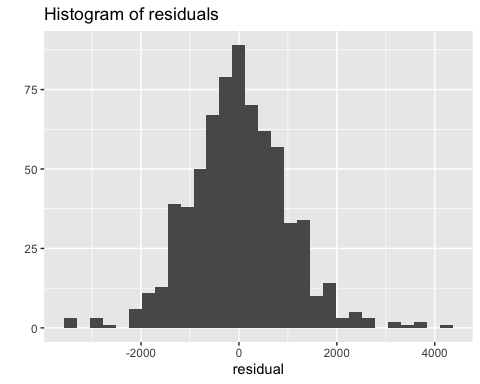
plot(mi.fitLR$finalModel, 3)



plot(mi.fitLR$finalModel, 4)



qplot(mi.fitLR$finalModel$residuals,  
 geom = "histogram",  
 bins = 30) +  
 labs(title = "Histogram of residuals",  
 x = "residual")



#plot\_grid(p1, p2, p3, p4, p5, ncol = 2, labels = "auto")

### RMSE for Test data

test.MonthlyIncome\_LR <- predict(mi.fitLR, newdata = mi.test)  
mi.fitLR.RMSE <- RMSE(test.MonthlyIncome\_LR, mi.test$MonthlyIncome)  
mi.fitLR.RMSE

## [1] 1165.628

RMSE(test.MonthlyIncome\_LR, mi.test$MonthlyIncome) / mean(mi.test$MonthlyIncome)

## [1] 0.1806066

This model is performing well within the $3,000 goal, so moving on to…

## Random Forest

### Train a Random Forest, tuning mtry and splitrule

set.seed(12)  
cv\_control <- trainControl(method="cv",   
 num = 5)  
  
mi.fitRF <- train(MonthlyIncome ~ .,   
 data = mi.train,   
 method = "ranger",   
 importance = "impurity",  
 trControl = cv\_control,  
 num.threads = 6,  
 num.trees = 100  
 )  
mi.fitRF

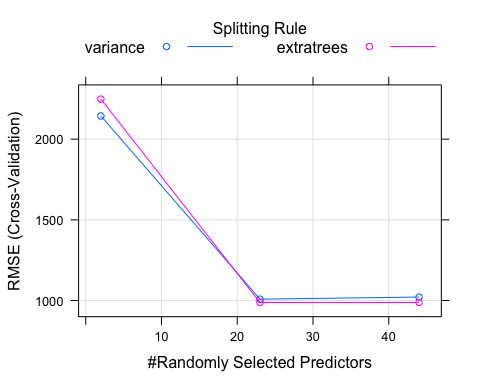
## Random Forest   
##   
## 696 samples  
## 30 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 557, 556, 557, 558, 556   
## Resampling results across tuning parameters:  
##   
## mtry splitrule RMSE Rsquared MAE   
## 2 variance 2143.9934 0.8796107 1663.7013  
## 2 extratrees 2248.3721 0.8924561 1763.8410  
## 23 variance 1008.2425 0.9512529 753.9700  
## 23 extratrees 987.7036 0.9535088 757.7323  
## 44 variance 1021.2030 0.9498785 760.8691  
## 44 extratrees 988.0584 0.9530846 749.7614  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 5  
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were mtry = 23, splitrule = extratrees  
## and min.node.size = 5.

### Performance on Training Set

summary(mi.fitRF)

## Length Class Mode   
## predictions 696 -none- numeric   
## num.trees 1 -none- numeric   
## num.independent.variables 1 -none- numeric   
## mtry 1 -none- numeric   
## min.node.size 1 -none- numeric   
## variable.importance 44 -none- numeric   
## prediction.error 1 -none- numeric   
## forest 7 ranger.forest list   
## splitrule 1 -none- character  
## num.random.splits 1 -none- numeric   
## treetype 1 -none- character  
## r.squared 1 -none- numeric   
## call 9 -none- call   
## importance.mode 1 -none- character  
## num.samples 1 -none- numeric   
## replace 1 -none- logical   
## xNames 44 -none- character  
## problemType 1 -none- character  
## tuneValue 3 data.frame list   
## obsLevels 1 -none- logical   
## param 3 -none- list

plot(mi.fitRF)



test.MonthlyIncome\_RF <- predict(mi.fitRF, newdata = mi.test)  
mi.fitRF.RMSE <- RMSE(test.MonthlyIncome\_RF, mi.test$MonthlyIncome)  
mi.fitRF.RMSE

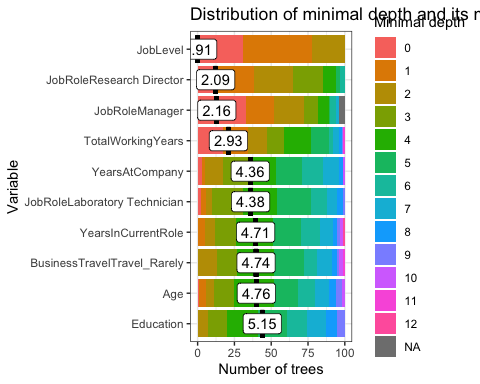
## [1] 1087.495

RMSE(test.MonthlyIncome\_RF, mi.test$MonthlyIncome) / mean(mi.test$MonthlyIncome)

## [1] 0.1685005

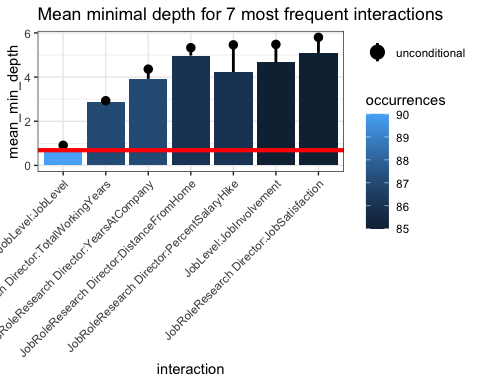
### Min Depth Distribution

mi.forest\_frame <- min\_depth\_distribution(mi.fitRF$finalModel)  
plot\_min\_depth\_distribution(mi.forest\_frame)

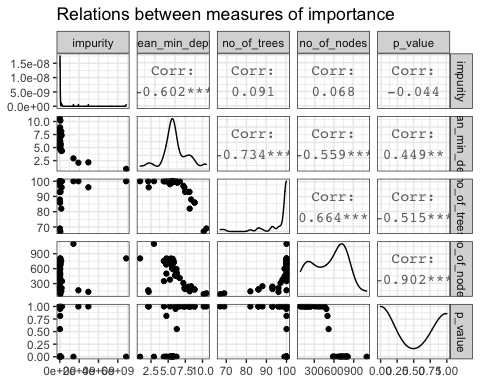


JobLevel has by far the most influence on MonthlyIncome, which makes sense given what we saw in the EDA. Really no surprises here.

### Mean minimal depth for most frequent interactions



Interaction between JobLevel and Age occurs most frequently in the random forest of trees. It’s interesting that JobRole Research Director and TotalWorkingYears interact moderately often and on average high up in their respective trees.



## Comparing MonthlyIncome Models

mi.compare <- data.frame(model = c("Linear Regression", "Random Forest"), RMSE = c(mi.fitLR.RMSE, mi.fitRF.RMSE))  
mi.compare

## model RMSE  
## 1 Linear Regression 1165.628  
## 2 Random Forest 1087.495

Both models perform well below the $3,000 RMSE threshold for this case study. Since Random Forest has slightly lower RMSE, Random Forest wins.

mi.winner <- mi.fitRF

# Attrition

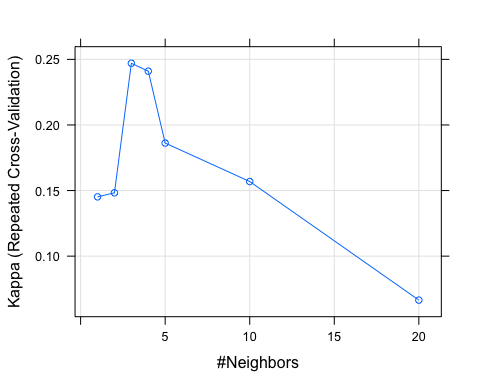
## KNN

For KNN, I’m optimizing for Kappa, which is more appropriate for imbalanced classes

set.seed(12)  
cv\_control <- trainControl(  
 method="repeatedcv",  
 repeats = 3,   
 number = 5,  
 classProbs = TRUE,  
 savePredictions = TRUE,  
 )  
  
at.fitKNN <- train(Attrition ~ .,   
 data = at.train,   
 method = "knn",   
 metric = "Kappa",  
 trControl = cv\_control,  
 preProcess = c("center","scale"),  
# tuneLength=20  
 tuneGrid = expand.grid(k = c(1:5, 10, 20))  
 )   
at.fitKNN

## k-Nearest Neighbors   
##   
## 696 samples  
## 32 predictor  
## 2 classes: 'No', 'Yes'   
##   
## Pre-processing: centered (53), scaled (53)   
## Resampling: Cross-Validated (5 fold, repeated 3 times)   
## Summary of sample sizes: 556, 557, 557, 557, 557, 557, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 1 0.8036382 0.14518601  
## 2 0.7993217 0.14829296  
## 3 0.8582288 0.24703291  
## 4 0.8596471 0.24100513  
## 5 0.8587050 0.18621525  
## 10 0.8615896 0.15687542  
## 20 0.8539294 0.06647419  
##   
## Kappa was used to select the optimal model using the largest value.  
## The final value used for the model was k = 3.

plot(at.fitKNN)



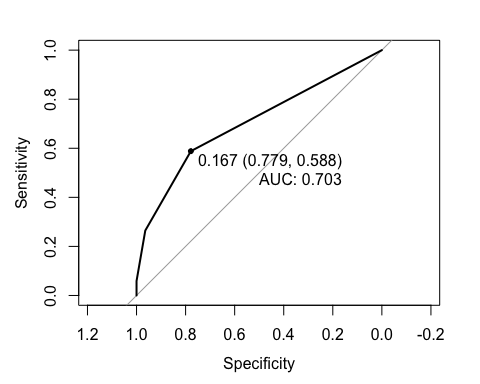
confusionMatrix(at.fitKNN)

## Cross-Validated (5 fold, repeated 3 times) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 82.7 12.1  
## Yes 2.1 3.2  
##   
## Accuracy (average) : 0.8582

at.fitKNN.predictions.raw <- predict(at.fitKNN, newdata = at.test, type="raw")  
at.fitKNN.predictions.prob <- predict(at.fitKNN, newdata = at.test, type="prob")  
confusionMatrix(at.fitKNN.predictions.raw, at.test$Attrition, positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 135 25  
## Yes 5 9  
##   
## Accuracy : 0.8276   
## 95% CI : (0.7631, 0.8805)  
## No Information Rate : 0.8046   
## P-Value [Acc > NIR] : 0.2552000   
##   
## Kappa : 0.2946   
##   
## Mcnemar's Test P-Value : 0.0005226   
##   
## Sensitivity : 0.26471   
## Specificity : 0.96429   
## Pos Pred Value : 0.64286   
## Neg Pred Value : 0.84375   
## Prevalence : 0.19540   
## Detection Rate : 0.05172   
## Detection Prevalence : 0.08046   
## Balanced Accuracy : 0.61450   
##   
## 'Positive' Class : Yes   
##

at.prediction.probabilities <- at.fitKNN.predictions.prob$Yes  
at.predicted.classes <- at.fitKNN.predictions.raw  
at.observed.classes <- at.test$Attrition  
  
at.res.roc <- roc(at.observed.classes, at.prediction.probabilities)  
plot.roc(at.res.roc, print.auc = TRUE, print.thres = "best")



# Get the best cutoff for balancing Sensitivity and Specificity  
cutoff <- coords(at.res.roc, "best", ret="threshold", transpose = FALSE)$threshold  
  
# Predict using the best cutoff and confirm with a Confusion Matrix  
at.predicted.classes.balanced <- factor(  
 ifelse( at.fitKNN.predictions.prob$Yes > cutoff, "Yes", "No"), levels=c("No","Yes"))  
at.fitKNN.cm <- confusionMatrix(at.predicted.classes.balanced, at.test$Attrition, positive="Yes")  
at.fitKNN.cm

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 109 14  
## Yes 31 20  
##   
## Accuracy : 0.7414   
## 95% CI : (0.6697, 0.8047)  
## No Information Rate : 0.8046   
## P-Value [Acc > NIR] : 0.98364   
##   
## Kappa : 0.3084   
##   
## Mcnemar's Test P-Value : 0.01707   
##   
## Sensitivity : 0.5882   
## Specificity : 0.7786   
## Pos Pred Value : 0.3922   
## Neg Pred Value : 0.8862   
## Prevalence : 0.1954   
## Detection Rate : 0.1149   
## Detection Prevalence : 0.2931   
## Balanced Accuracy : 0.6834   
##   
## 'Positive' Class : Yes   
##

# Random Forest

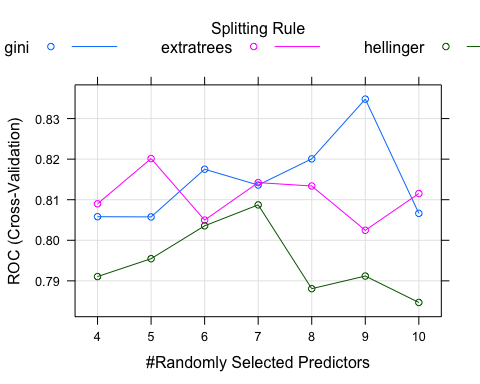
# Train a Random Forest, tuning mtry and splitrule

set.seed(12)  
cv\_control <- trainControl(method="cv",   
 classProbs = TRUE,  
 savePredictions = TRUE,  
 summaryFunction = twoClassSummary,  
 num = 5)  
  
rf\_grid <- expand.grid(  
 mtry = 4:10,  
 splitrule = c("gini","extratrees", "hellinger"),  
 min.node.size = c(1)  
)  
  
at.fitRF <- train(Attrition ~ .,   
 data = at.train,   
 method = "ranger",   
 metric = "ROC",  
 importance = "impurity",  
 trControl = cv\_control,  
 num.threads = 6,  
 num.trees = 100,  
 tuneGrid=rf\_grid)   
at.fitRF

## Random Forest   
##   
## 696 samples  
## 32 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 556, 557, 557, 557, 557   
## Resampling results across tuning parameters:  
##   
## mtry splitrule ROC Sens Spec   
## 4 gini 0.8058111 0.9966102 0.1134199  
## 4 extratrees 0.8089919 0.9983051 0.1415584  
## 4 hellinger 0.7910522 0.9983051 0.1134199  
## 5 gini 0.8057616 0.9949153 0.1229437  
## 5 extratrees 0.8201372 0.9949153 0.1225108  
## 5 hellinger 0.7954619 0.9966102 0.1411255  
## 6 gini 0.8175141 0.9949153 0.1229437  
## 6 extratrees 0.8049784 0.9966102 0.1406926  
## 6 hellinger 0.8035641 0.9949153 0.1324675  
## 7 gini 0.8136107 0.9949153 0.1696970  
## 7 extratrees 0.8142142 0.9983051 0.1506494  
## 7 hellinger 0.8087295 0.9915254 0.1510823  
## 8 gini 0.8200712 0.9949153 0.1601732  
## 8 extratrees 0.8133759 0.9949153 0.1783550  
## 8 hellinger 0.7880751 0.9898305 0.1692641  
## 9 gini 0.8347953 0.9932203 0.1787879  
## 9 extratrees 0.8024598 0.9915254 0.2073593  
## 9 hellinger 0.7911824 0.9898305 0.1696970  
## 10 gini 0.8066347 0.9915254 0.1974026  
## 10 extratrees 0.8115196 0.9915254 0.2164502  
## 10 hellinger 0.7846706 0.9932203 0.1701299  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 1  
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 9, splitrule = gini  
## and min.node.size = 1.

# Performance on Training Set

plot(at.fitRF)

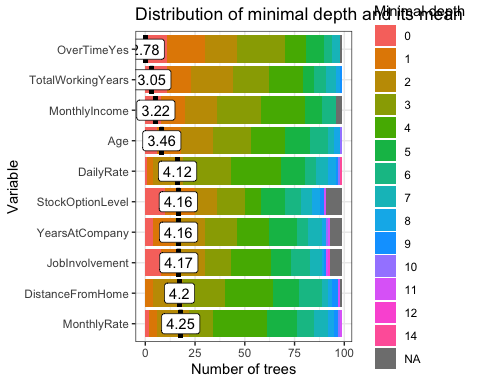


confusionMatrix(at.fitRF)

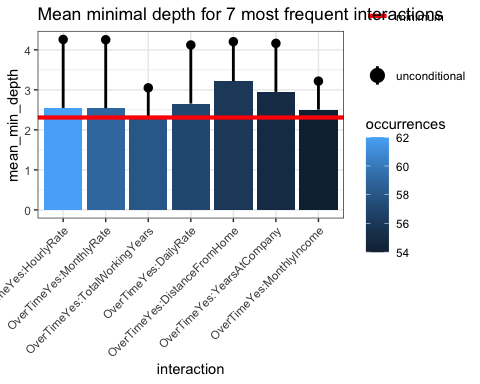
## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 84.2 12.5  
## Yes 0.6 2.7  
##   
## Accuracy (average) : 0.8693

## Min Depth Distribution

at.forest\_frame <- min\_depth\_distribution(at.fitRF$finalModel)  
plot\_min\_depth\_distribution(at.forest\_frame)

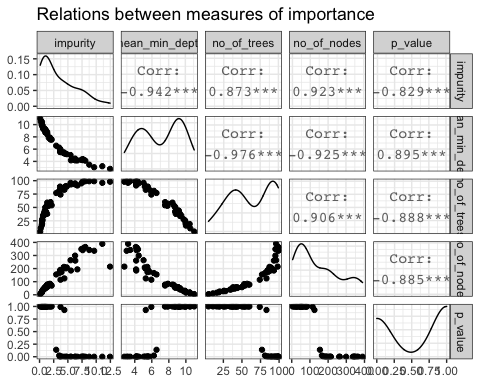
 Overtime is clearly the most influential variable on its own, but let’s look at interactions…

### Mean minimal depth for most frequent interactions



An interesting observation is that HourlyRate on its own is one of the less influential variables, but when combined with OverTimeYes, it becomes the most influential interaction.

### Other measures of importance



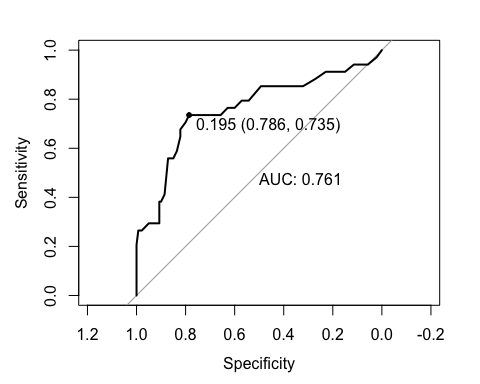
# Performance on Test Set

at.fitRF.predictions.raw <- predict(at.fitRF, newdata = at.test, type="raw")  
at.fitRF.predictions.prob <- predict(at.fitRF, newdata = at.test, type="prob")  
confusionMatrix(at.fitRF.predictions.raw, at.test$Attrition, positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 140 29  
## Yes 0 5  
##   
## Accuracy : 0.8333   
## 95% CI : (0.7695, 0.8854)  
## No Information Rate : 0.8046   
## P-Value [Acc > NIR] : 0.1962   
##   
## Kappa : 0.2172   
##   
## Mcnemar's Test P-Value : 1.999e-07   
##   
## Sensitivity : 0.14706   
## Specificity : 1.00000   
## Pos Pred Value : 1.00000   
## Neg Pred Value : 0.82840   
## Prevalence : 0.19540   
## Detection Rate : 0.02874   
## Detection Prevalence : 0.02874   
## Balanced Accuracy : 0.57353   
##   
## 'Positive' Class : Yes   
##

# ROC Curve and Optimal Cutoff

at.prediction.probabilities <- at.fitRF.predictions.prob$Yes  
at.predicted.classes <- at.fitRF.predictions.raw  
at.observed.classes <- at.test$Attrition  
  
# Compute roc  
at.res.roc <- roc(at.observed.classes, at.prediction.probabilities)  
plot.roc(at.res.roc, print.auc = TRUE, print.thres = "best")



# Get the best cutoff for balancing Sensitivity and Specificity  
at.cutoff.randomforest <- coords(at.res.roc, "best", ret="threshold", transpose = FALSE)$threshold  
  
# Predict using the best cutoff and confirm with a Confusion Matrix  
at.predicted.classes.balanced <- factor(  
 ifelse( at.fitRF.predictions.prob$Yes > at.cutoff.randomforest, "Yes", "No"), levels=c("No","Yes"))  
at.fitRF.cm <- confusionMatrix(at.predicted.classes.balanced, at.test$Attrition, positive="Yes")  
at.fitRF.cm

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 110 9  
## Yes 30 25  
##   
## Accuracy : 0.7759   
## 95% CI : (0.7066, 0.8355)  
## No Information Rate : 0.8046   
## P-Value [Acc > NIR] : 0.853196   
##   
## Kappa : 0.4223   
##   
## Mcnemar's Test P-Value : 0.001362   
##   
## Sensitivity : 0.7353   
## Specificity : 0.7857   
## Pos Pred Value : 0.4545   
## Neg Pred Value : 0.9244   
## Prevalence : 0.1954   
## Detection Rate : 0.1437   
## Detection Prevalence : 0.3161   
## Balanced Accuracy : 0.7605   
##   
## 'Positive' Class : Yes   
##

# Accuracy Model Comparison

Model <- c("KNN", "Random Forest")  
Accuracy <- c(at.fitKNN.cm$overall['Accuracy'], at.fitRF.cm$overall['Accuracy'])  
Kappa <- c(at.fitKNN.cm$overall['Kappa'], at.fitRF.cm$overall['Kappa'])  
Sensitivity <- c(at.fitKNN.cm$byClass['Sensitivity'], at.fitRF.cm$byClass['Sensitivity'])  
Specificity <- c(at.fitKNN.cm$byClass['Specificity'], at.fitRF.cm$byClass['Specificity'])  
at.compare <- data.frame(Model, Accuracy, Kappa, Sensitivity, Specificity)  
at.compare

## Model Accuracy Kappa Sensitivity Specificity  
## 1 KNN 0.7413793 0.3084261 0.5882353 0.7785714  
## 2 Random Forest 0.7758621 0.4222714 0.7352941 0.7857143

KNN is overall more accurate, but Random Forest has slightly higher Kappa and more Sensitivity, while being a bit less Specific. Also, KNN dipped below 60 on Sensitivity, so for the purposes of this case study, which requires both Sensitivity and Specificity to be more than 60, Random Forest wins.

at.winner <- at.fitRF  
at.winner.cutoff <- at.cutoff.randomforest

# Comp Set: No Attrition

## Load the No Attrition comp set and clean it the same way we did for modeling

dfNoAttrition <- load.and.clean.for.attrition("doc/CaseStudy2CompSet No Attrition.csv", removeID = FALSE)

## Predict Attrition using our best model from the training above

# Use the best model to get predicted probabilities for each class  
dfNoAttrition.predictions.prob <- predict(at.winner, newdata = dfNoAttrition, type="prob")  
  
# Use the most balanced cutoff on the predicted probabilities to get the Attrition values  
dfNoAttrition.predictions.class <- factor(  
 ifelse( dfNoAttrition.predictions.prob$Yes > at.winner.cutoff, "Yes", "No"), levels=c("No","Yes"))

## Save

dfNoAttritionSubmission <- data.frame(ID = dfNoAttrition$ID, Attrition = dfNoAttrition.predictions.class)  
write.csv(dfNoAttritionSubmission, "Case2PredictionsLichti Attrition.csv")

# Comp Set: No Salary

## Load the No Salary comp set and clean it the same way we did for modeling

dfNoSalary <- load.and.clean.for.monthly\_income("doc/CaseStudy2CompSet No Salary.csv", removeID = FALSE)

## Predict Salary

dfNoSalary$MonthlyIncome <- predict(mi.winner, newdata = dfNoSalary)

## Save

dfNoSalarySubmission <- data.frame(ID = dfNoSalary$ID, MonthlyIncome = dfNoSalary$MonthlyIncome)  
write.csv(dfNoSalarySubmission, "Case2PredictionsLichti Salary.csv")