Team-X Final Project: Attrition

## Introduction

*TeamX* This project was completed by:

* Amy McVicar
* Angela Garcia
* Jennifer Mead
* Jonathan Lee

Businesses lose money when they lose employees. Employee attrition impacts businesses due to the costs of hiring and training new employees. Because of this, data-driven HR departments use data to identify who is likely to quit and to find trends in what factors influence quitting decisions, such as particular departments or locations. [1]

Companies have always been concerned about attrition, but “in many industries the cost of losing good workers is rising” [2]. Exact numbers vary by industry. For example, “[estimates] of annual turnover among U.S. salespeople run as high as 27%—twice the rate in the overall labor force.” [3]

A high attrition rate adds up: “U.S. firms spend $15 billion a year training salespeople and another $800 billion on incentives, and attrition reduces the return on those investments.” [4] In some cases, the cost of losing an employee can be as much as twice their yearly salary. [5]

When employees see other employees leave, attrition can increase. “In settings with high voluntary turnover, employees often lose faith in the company’s strategic direction (because they see others jumping ship), and they tend to be more aware of outside job opportunities, partly because their networks include former colleagues who recently defected. And when there’s lots of involuntary turnover, employees may lack trust in managers, feel little job security, and move on.” [6]

Those costs add up. “It takes an average of 24 days to fill a job, costing employers up to $4,000 per hire– maybe more, depending on your industry.”[7]

Another study “estimates that 42 million, or one in four, employees will leave their jobs in 2018, and that nearly 77 percent, or three-fourths, of that turnover could be prevented by employers.”[8]

Indicators to look for Researchers have found many factors that can be used to identify an increased likelihood of quitting. One study found that these “… include leaving work early, showing less focus or effort, and being reluctant to commit to long-term assignments.” [9]

Another study found that among people who left within the first six months, common issues were: not having clear priorities, a lack of effective training, and not feeling recognized for their contributions. [10]

Some research has been done on specific groups. Executives may have different motivators than sales people. One study identified key factors for executives leaving jobs in less than a year, including pay, a work culture that doesn’t recognize performance, and a lack of synergy among bosses, peers, and direct reports. [11]

Because there are many potential factors that influence voluntary attrition and because there is known variation between industries, roles, and companies, it is useful for companies to analyze their own data to determine patterns in their attrition.

## Analysis and Models

This analysis looks at data from IBM that shows common attrition factors for a fictional company.

Analysis will include using a variety of visualization and machine learning methods and then comparing the results. Combining methods helps to reduce bias [12] and gives a more comprehensive view of the data.

### About the data

Download the data from <https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset>

Before running models on the data, the following steps were performed:

* identify variables to remove because the data is bad or not useful
* set appropriate types for each column (e.g. factor, numeric)
* visualize the variables to give a sense of where to focus analysis
* look for associations/correlations between variables
* perform data transformations for each method, such as creating transactions before using ARM and converting values to numbers before using k-means

#### Load the data

# load in the data  
HR\_original <- read.csv("http://www.creativecubecompany.com/syracuse/ist707/Attrition\_ORIGINAL.csv", fileEncoding ="UTF-8-BOM")

#### Clean the data

Look at the range and typical values for all variables to identify if any should be eliminated due to not being useful.

HR\_clean <- HR\_original  
summary(HR\_clean)

## 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 Min. :1.000   
## Research & Development:961 1st Qu.: 2.000 1st Qu.:2.000   
## Sales :446 Median : 7.000 Median :3.000   
## Mean : 9.193 Mean :2.913   
## 3rd Qu.:14.000 3rd Qu.:4.000   
## Max. :29.000 Max. :5.000   
##   
## EducationField EmployeeCount EmployeeNumber EnvironmentSatisfaction  
## Human Resources : 27 Min. :1 Min. : 1.0 Min. :1.000   
## Life Sciences :606 1st Qu.:1 1st Qu.: 491.2 1st Qu.:2.000   
## Marketing :159 Median :1 Median :1020.5 Median :3.000   
## Medical :464 Mean :1 Mean :1024.9 Mean :2.722   
## Other : 82 3rd Qu.:1 3rd Qu.:1555.8 3rd Qu.:4.000   
## Technical Degree:132 Max. :1 Max. :2068.0 Max. :4.000   
##   
## Gender HourlyRate JobInvolvement JobLevel   
## Female:588 Min. : 30.00 Min. :1.00 Min. :1.000   
## Male :882 1st Qu.: 48.00 1st Qu.:2.00 1st Qu.:1.000   
## Median : 66.00 Median :3.00 Median :2.000   
## Mean : 65.89 Mean :2.73 Mean :2.064   
## 3rd Qu.: 83.75 3rd Qu.:3.00 3rd Qu.:3.000   
## Max. :100.00 Max. :4.00 Max. :5.000   
##   
## JobRole JobSatisfaction MaritalStatus MonthlyIncome   
## Sales Executive :326 Min. :1.000 Divorced:327 Min. : 1009   
## Research Scientist :292 1st Qu.:2.000 Married :673 1st Qu.: 2911   
## Laboratory Technician :259 Median :3.000 Single :470 Median : 4919   
## Manufacturing Director :145 Mean :2.729 Mean : 6503   
## Healthcare Representative:131 3rd Qu.:4.000 3rd Qu.: 8379   
## Manager :102 Max. :4.000 Max. :19999   
## (Other) :215   
## MonthlyRate NumCompaniesWorked Over18 OverTime PercentSalaryHike  
## Min. : 2094 Min. :0.000 Y:1470 No :1054 Min. :11.00   
## 1st Qu.: 8047 1st Qu.:1.000 Yes: 416 1st Qu.:12.00   
## Median :14236 Median :2.000 Median :14.00   
## Mean :14313 Mean :2.693 Mean :15.21   
## 3rd Qu.:20462 3rd Qu.:4.000 3rd Qu.:18.00   
## Max. :26999 Max. :9.000 Max. :25.00   
##   
## PerformanceRating RelationshipSatisfaction StandardHours StockOptionLevel  
## Min. :3.000 Min. :1.000 Min. :80 Min. :0.0000   
## 1st Qu.:3.000 1st Qu.:2.000 1st Qu.:80 1st Qu.:0.0000   
## Median :3.000 Median :3.000 Median :80 Median :1.0000   
## Mean :3.154 Mean :2.712 Mean :80 Mean :0.7939   
## 3rd Qu.:3.000 3rd Qu.:4.000 3rd Qu.:80 3rd Qu.:1.0000   
## Max. :4.000 Max. :4.000 Max. :80 Max. :3.0000   
##   
## TotalWorkingYears TrainingTimesLastYear WorkLifeBalance YearsAtCompany   
## Min. : 0.00 Min. :0.000 Min. :1.000 Min. : 0.000   
## 1st Qu.: 6.00 1st Qu.:2.000 1st Qu.:2.000 1st Qu.: 3.000   
## Median :10.00 Median :3.000 Median :3.000 Median : 5.000   
## Mean :11.28 Mean :2.799 Mean :2.761 Mean : 7.008   
## 3rd Qu.:15.00 3rd Qu.:3.000 3rd Qu.:3.000 3rd Qu.: 9.000   
## Max. :40.00 Max. :6.000 Max. :4.000 Max. :40.000   
##   
## YearsInCurrentRole YearsSinceLastPromotion YearsWithCurrManager  
## Min. : 0.000 Min. : 0.000 Min. : 0.000   
## 1st Qu.: 2.000 1st Qu.: 0.000 1st Qu.: 2.000   
## Median : 3.000 Median : 1.000 Median : 3.000   
## Mean : 4.229 Mean : 2.188 Mean : 4.123   
## 3rd Qu.: 7.000 3rd Qu.: 3.000 3rd Qu.: 7.000   
## Max. :18.000 Max. :15.000 Max. :17.000   
##

str(HR\_clean)

[structure removed for ease of reading]

Actions:

* remove EmployeeCount because it is always 1
* remove Over18 because it is always Y
* remove StandardHours because it is always 80
* change the name of i..Age to fix a typo

# reference-- drop columns by name: https://stackoverflow.com/questions/5234117/how-to-drop-columns-by-name-in-a-data-frame  
# reference -- move column to the first column: https://stackoverflow.com/questions/22286419/move-a-column-to-first-position-in-a-data-frame  
  
library(dplyr)

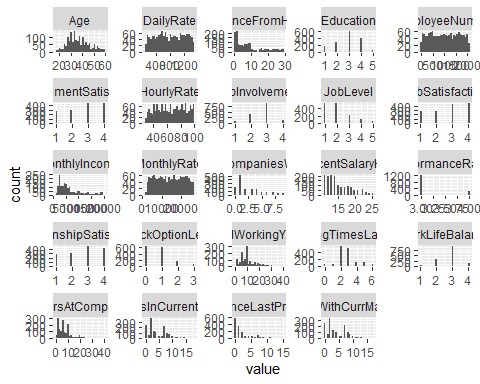
HR\_clean <- subset(HR\_clean, select=-c(EmployeeCount, StandardHours, Over18))  
  
HR\_clean <- HR\_clean %>%  
 select(EmployeeNumber, everything())  
head(HR\_clean, 10)

[head removed for ease of reading]

Look at histograms of all numeric variables to identify which should be categorical instead

# reference-- histogram of all variables: https://drsimonj.svbtle.com/quick-plot-of-all-variables  
library(purrr)  
library(tidyr)  
library(ggplot2)  
  
HR\_clean %>%  
 keep(is.numeric) %>%   
 gather() %>%   
 ggplot(aes(value)) +  
 facet\_wrap(~ key, scales = "free") +  
 geom\_histogram()

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



Histograms with parallel lines instead of of distributions with close bins are typically factors. It looks like the following columns are actually factors instead of integers. Some models need numerical inputs and some need factors, but for the moment they should be converted.

Numerical columns that should be factors:

* Education
* EnvironmentSatisfaction
* JobInvolvement
* JobLevel
* JobSatisfaction
* PerformanceRating
* RelationshipSatisfaction
* StockOptionLevel
* WorkLifeBalance

HR\_clean$Education <- as.factor(HR\_clean$Education)  
HR\_clean$EnvironmentSatisfaction <- as.factor(HR\_clean$EnvironmentSatisfaction)  
HR\_clean$JobInvolvement <- as.factor(HR\_clean$JobInvolvement)  
HR\_clean$JobLevel <- as.factor(HR\_clean$JobLevel)  
HR\_clean$JobSatisfaction <- as.factor(HR\_clean$JobSatisfaction)  
HR\_clean$PerformanceRating <- as.factor(HR\_clean$PerformanceRating)  
HR\_clean$RelationshipSatisfaction <- as.factor(HR\_clean$RelationshipSatisfaction)  
HR\_clean$StockOptionLevel <- as.factor(HR\_clean$StockOptionLevel)  
HR\_clean$WorkLifeBalance <- as.factor(HR\_clean$WorkLifeBalance)  
  
head(HR\_clean)

[head removed for ease of reading]

Check for blanks

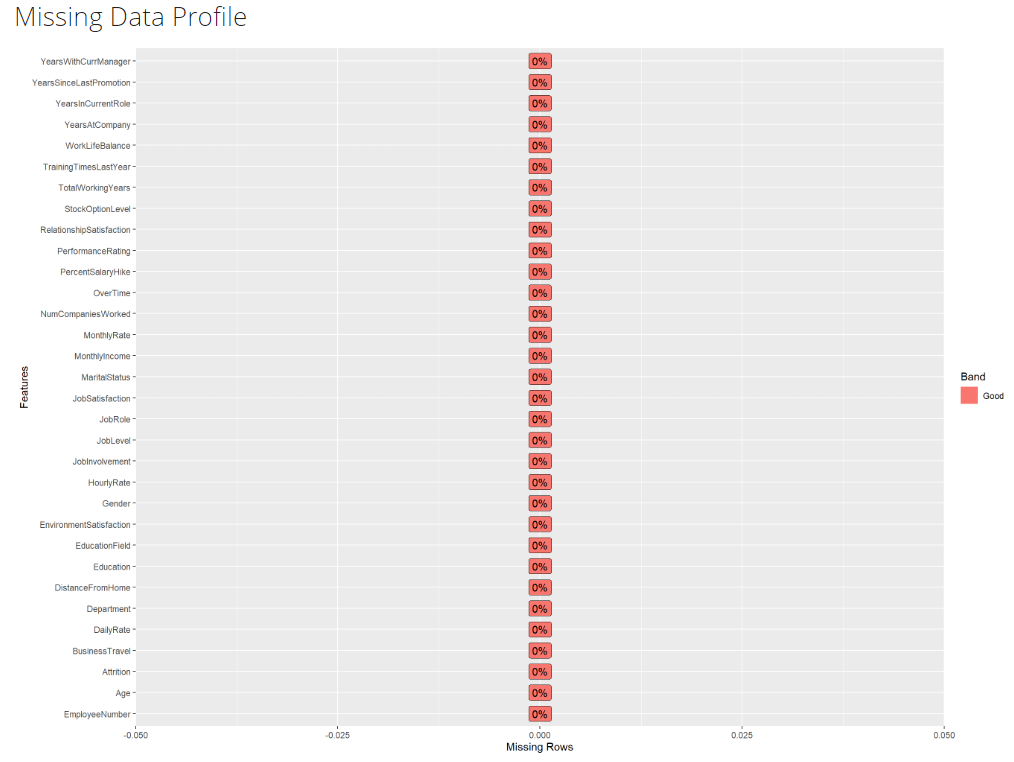
#reference -- checking for blanks: https://stackoverflow.com/questions/40715508/r-count-cells-with-missing-values-across-each-row  
  
colSums(is.na(HR\_clean) | HR\_clean == "" | HR\_clean == " ")

## EmployeeNumber Age Attrition   
## 0 0 0   
## BusinessTravel DailyRate Department   
## 0 0 0   
## DistanceFromHome Education EducationField   
## 0 0 0   
## EnvironmentSatisfaction Gender HourlyRate   
## 0 0 0   
## JobInvolvement JobLevel JobRole   
## 0 0 0   
## JobSatisfaction MaritalStatus MonthlyIncome   
## 0 0 0   
## MonthlyRate NumCompaniesWorked OverTime   
## 0 0 0   
## PercentSalaryHike PerformanceRating RelationshipSatisfaction   
## 0 0 0   
## StockOptionLevel TotalWorkingYears TrainingTimesLastYear   
## 0 0 0   
## WorkLifeBalance YearsAtCompany YearsInCurrentRole   
## 0 0 0   
## YearsSinceLastPromotion YearsWithCurrManager   
## 0 0

#### Visualize the variables

There are 32 variables in total. We can check again for any missing variables, and there are none.

if("DataExplorer" %in% rownames(installed.packages()) == FALSE) {install.packages('DataExplorer') }  
library(DataExplorer)  
HR\_eda <- HR\_clean  
plot\_str(HR\_eda)  
plot\_missing(HR\_eda)



From correlating the attributes we can see pockets of correlation.

Most notably are:

Years with Current Manager

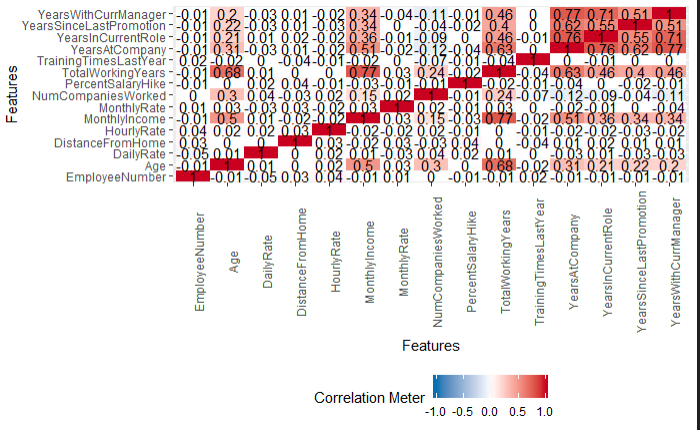
Years Since Last Promotion

Years in Current Role

Years at Company

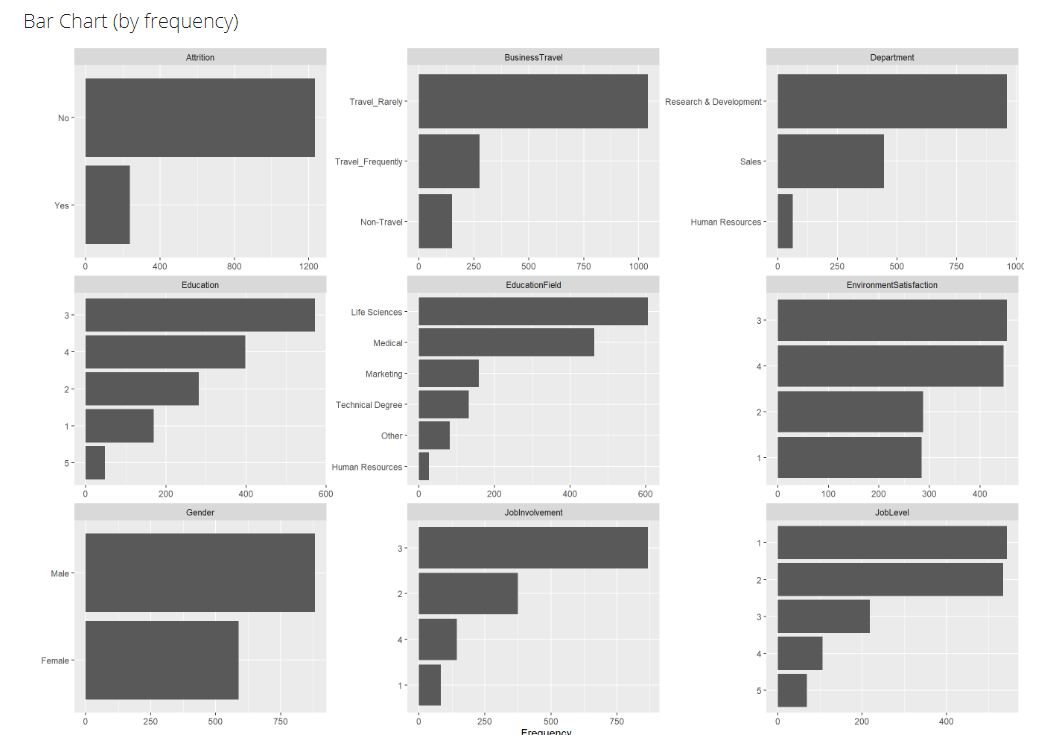
And no surprise, these correlate with Age, Income, and Total Working Years.

plot\_correlation(HR\_eda, type = 'continuous')



Simple barcharts of the attributes show us some interesting facts that we can use for deeper analysis. For example, most of the universe is ‘no’ to attrition. The Education Field and Department are limited in the selections available. This might help us understand the context of the findings of models. For example, there are only three department types (R&D, Sales, & HR). We might find that the weight of this attribute in models may only be relevant to this limited dataset and not as applicable to datasets that are more representative of real organizations. This is something we might not notice without this simple exploratory examination of the data first.

plot\_bar(HR\_eda)



[results removed for brevity]

#create\_report(HR\_eda)

Each variable, except EmployeeNumber, in the data set is examined for significant variance in the attrition yes versus no segments using simple analysis and plotting.

plot(HR\_eda$Attrition, HR\_eda$Age, main = "Age", ylab = "Age", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$BusinessTravel, main = "Business Travel", ylab = "Age", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$DailyRate, main = "Daily Rate", ylab = "Daily Rate", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$Department, main = "Department", ylab = "Department", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$DistanceFromHome, main = "Distance From Home", ylab = "Distance From Home", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$Education, main = "Education", ylab = "Education", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$EducationField, main = "Education Field", ylab = "Education Field", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$EnvironmentSatisfaction, main = "Environmental Satisfaction", ylab = "Environmental Satisfaction", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$Gender, main = "Gender", ylab = "Gender", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$HourlyRate, main = "Hourly Rate", ylab = "Hourly Rate", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$JobInvolvement, main = "Job Involvment", ylab = "Job Involvement", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$JobLevel, main = "Job Level", ylab = "Job Level", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$JobRole, main = "Job Role", ylab = "Job Role", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$JobSatisfaction, main = "Job Satisfaction", ylab = "Job Satisfaction", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$MaritalStatus, main = "Marital Status", ylab = "Marital Status", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$MonthlyIncome, main = "Monthly Income", ylab = "Monthly Income", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$MonthlyRate, main = "Monthly Rate", ylab = "Monthly Rate", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$NumCompaniesWorked, main = "Num Companies Worked", ylab = "Num Companies Worked", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$OverTime, main = "Over Time", ylab = "Over Time", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$PercentSalaryHike, main = "Percent Salary Hike", ylab = "Percent Salary Hike", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$PerformanceRating, main = "Performance Rating", ylab = "Performance Rating", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$RelationshipSatisfaction, main = "Relationship Satisfaction", ylab = "Relationship Satisfaction", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$StockOptionLevel, main = "Stock Option Level", ylab = "Stock Option Level", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$TotalWorkingYears, main = "Total Working Years", ylab = "Total Working Years", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$TrainingTimesLastYear, main = "Training Times Last Year", ylab = "Training Times Last Year", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$WorkLifeBalance, main = "Work Life Balance", ylab = "Work Life Balance", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$YearsAtCompany, main = "Years at Company", ylab = "Years at Company", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$YearsInCurrentRole, main = "Years in Current Role", ylab = "Years in Current Role", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$YearsSinceLastPromotion, main = "Years Since Last Promotion", ylab = "Years Since Last Promotion", xlab = "Attrition")

plot(HR\_eda$Attrition, HR\_eda$YearsWithCurrManager, main = "Years With Current Manager", ylab = "Years With Current Manager", xlab = "Attrition")

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |

On visual inspection the following variables appear to have a significant difference in the attrition yes and no segments:

* EnvironmentalSatisfaction
* JobInvolvement
* JobLevel
* JobRole
* JobSatisfaction
* MaritalStatus
* MonthlyIncome
* NumCompaniesWorked
* OverTime
* RelationshipSatisfaction

On initial visual analysis and inspection, the following attributes may have significance:

* StopOptionLevel
* TotalWorkingYears
* TrainingTimesLastYear
* WorkLifeBalance
* YearsAtCompany
* YearsInCurrentCompany
* YearsInCurrentRole
* YearsWithCurrentManager

Additionally, initial inspection shows that more than a few attributes appear to be highly correlated with each other. This information may be used for further analysis and refining the attributes used in models for simplification.

#Install packages if they dont exist  
  
library(formattable)  
library(gridExtra)  
library(grid)  
library(corrplot)  
library(rquery)  
library(GoodmanKruskal)

# Data Transformation  
  
# Data Assessment  
HR\_linear<-HR\_clean  
  
#Create Categories for numeric values with high number of records (Based on Percentiles)  
  
## Categoric Age  
  
# Age Percentiles  
Percentile\_00 = min(HR\_linear$Age)  
Percentile\_33 = quantile(HR\_linear$Age, 0.33333)  
Percentile\_67 = quantile(HR\_linear$Age, 0.66667)  
Percentile\_100 = max(HR\_linear$Age)  
  
# Values  
HR.BindA = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.BindA)[[2]] = "Value"  
#HR.BindA  
  
#Age:   
HR\_linear$AgeRange[HR\_linear$Age >= Percentile\_00 & HR\_linear$Age < Percentile\_33] = "Lower\_Range"  
HR\_linear$AgeRange[HR\_linear$Age >= Percentile\_33 & HR\_linear$Age < Percentile\_67] = "Mid\_Range"  
HR\_linear$AgeRange[HR\_linear$Age >= Percentile\_67 & HR\_linear$Age <= Percentile\_100] = "Higher\_Range"  
  
## Categoric Hourly Rate  
  
# Hourly Rate Percentiles  
Percentile\_00 = min(HR\_linear$HourlyRate)  
Percentile\_33 = quantile(HR\_linear$HourlyRate, 0.33333)  
Percentile\_67 = quantile(HR\_linear$HourlyRate, 0.66667)  
Percentile\_100 = max(HR\_linear$HourlyRate)  
  
# Values  
HR.BindH = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.BindH)[[2]] = "Value"  
#HR.BindH  
  
#Hourly Rate Ranges:   
HR\_linear$HourlyRateRange[HR\_linear$HourlyRate >= Percentile\_00 & HR\_linear$HourlyRate < Percentile\_33] = "Low\_Range"  
HR\_linear$HourlyRateRange[HR\_linear$HourlyRate >= Percentile\_33 & HR\_linear$HourlyRate < Percentile\_67] = "Mid\_Range"  
HR\_linear$HourlyRateRange[HR\_linear$HourlyRate >= Percentile\_67 & HR\_linear$HourlyRate <= Percentile\_100] = "High\_Range"  
  
## Categoric Daily Rate  
  
# Daily Rate Percentiles  
Percentile\_00 = min(HR\_linear$DailyRate)  
Percentile\_33 = quantile(HR\_linear$DailyRate, 0.33333)  
Percentile\_67 = quantile(HR\_linear$DailyRate, 0.66667)  
Percentile\_100 = max(HR\_linear$DailyRate)  
  
# Values  
HR.BindDR = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.BindDR)[[2]] = "Value"  
#HR.BindDR  
  
# Daily Rate Ranges:   
HR\_linear$DailyRateRange[HR\_linear$DailyRate >= Percentile\_00 & HR\_linear$DailyRate < Percentile\_33] = "Low\_Range"  
HR\_linear$DailyRateRange[HR\_linear$DailyRate >= Percentile\_33 & HR\_linear$DailyRate < Percentile\_67] = "Mid\_Range"  
HR\_linear$DailyRateRange[HR\_linear$DailyRate >= Percentile\_67 & HR\_linear$DailyRate <= Percentile\_100] = "High\_Range"  
  
  
## Categoric Monthly Rate  
  
# Monthly Rate Percentiles  
Percentile\_00 = min(HR\_linear$MonthlyRate)  
Percentile\_33 = quantile(HR\_linear$MonthlyRate, 0.33333)  
Percentile\_67 = quantile(HR\_linear$MonthlyRate, 0.66667)  
Percentile\_100 = max(HR\_linear$MonthlyRate)  
  
# Values  
HR.BindMR = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.BindMR)[[2]] = "Value"  
#HR.BindMR  
  
# Monthly Rate Level  
HR\_linear$MonthRateLevel[HR\_linear$MonthlyRate >= Percentile\_00 & HR\_linear$MonthlyRate < Percentile\_33] = "Low\_Income"  
HR\_linear$MonthRateLevel[HR\_linear$MonthlyRate >= Percentile\_33 & HR\_linear$MonthlyRate < Percentile\_67] = "Mid\_Income"  
HR\_linear$MonthRateLevel[HR\_linear$MonthlyRate >= Percentile\_67 & HR\_linear$MonthlyRate <= Percentile\_100] = "High\_Income"  
  
# Categoric Monthly Income  
  
# Monthly Income Percentiles  
Percentile\_00 = min(HR\_linear$MonthlyIncome)  
Percentile\_33 = quantile(HR\_linear$MonthlyIncome, 0.33333)  
Percentile\_67 = quantile(HR\_linear$MonthlyIncome, 0.66667)  
Percentile\_100 = max(HR\_linear$MonthlyIncome)  
  
# Values  
HR.BindI = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.BindI)[[2]] = "Value"  
#HR.BindI  
  
# Monthly Income Level  
HR\_linear$MonthIncomeLevel[HR\_linear$MonthlyIncome >= Percentile\_00 & HR\_linear$MonthlyIncome < Percentile\_33] = "Low\_Income"  
HR\_linear$MonthIncomeLevel[HR\_linear$MonthlyIncome >= Percentile\_33 & HR\_linear$MonthlyIncome < Percentile\_67] = "Mid\_Income"  
HR\_linear$MonthIncomeLevel[HR\_linear$MonthlyIncome >= Percentile\_67 & HR\_linear$MonthlyIncome <= Percentile\_100] = "High\_Income"  
  
# Categoric Distance From Home  
  
# Distance From Home Percentiles  
Percentile\_00 = min(HR\_linear$DistanceFromHome)  
Percentile\_33 = quantile(HR\_linear$DistanceFromHome, 0.33333)  
Percentile\_67 = quantile(HR\_linear$DistanceFromHome, 0.66667)  
Percentile\_100 = max(HR\_linear$DistanceFromHome)  
  
# Values  
HR.BindD = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.BindD)[[2]] = "Value"  
#HR.BindD  
  
# Distance From Home Ranges:   
HR\_linear$DistHomeRange[HR\_linear$DistanceFromHome >= Percentile\_00 & HR\_linear$DistanceFromHome < Percentile\_33] = "Low\_Distance"  
HR\_linear$DistHomeRange[HR\_linear$DistanceFromHome >= Percentile\_33 & HR\_linear$DistanceFromHome < Percentile\_67] = "Mid\_Distance"  
HR\_linear$DistHomeRange[HR\_linear$DistanceFromHome >= Percentile\_67 & HR\_linear$DistanceFromHome <= Percentile\_100] = "High\_Distance"  
  
# Categoric Number of Companies Worked  
  
# Number of Companies worked Percentiles  
Percentile\_00 = min(HR\_linear$NumCompaniesWorked)  
Percentile\_33 = quantile(HR\_linear$NumCompaniesWorked, 0.33333)  
Percentile\_67 = quantile(HR\_linear$NumCompaniesWorked, 0.66667)  
Percentile\_100 = max(HR\_linear$NumCompaniesWorked)  
  
# Values  
HR.BindC = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.BindC)[[2]] = "Value"  
#HR.BindC  
  
# Number of Companies worked Ranges:   
HR\_linear$NumCompWorked[HR\_linear$NumCompaniesWorked >= Percentile\_00 & HR\_linear$NumCompaniesWorked < Percentile\_33] = "Low\_Number"  
HR\_linear$NumCompWorked[HR\_linear$NumCompaniesWorked >= Percentile\_33 & HR\_linear$NumCompaniesWorked < Percentile\_67] = "Mid\_Number"  
HR\_linear$NumCompWorked[HR\_linear$NumCompaniesWorked >= Percentile\_67 & HR\_linear$NumCompaniesWorked <= Percentile\_100] = "High\_Number"  
  
# Categoric Salary Increase  
  
# Salary Increase Percentiles  
Percentile\_00 = min(HR\_linear$PercentSalaryHike)  
Percentile\_33 = quantile(HR\_linear$PercentSalaryHike, 0.33333)  
Percentile\_67 = quantile(HR\_linear$PercentSalaryHike, 0.66667)  
Percentile\_100 = max(HR\_linear$PercentSalaryHike)  
  
# Values  
HR.BindS = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.BindS)[[2]] = "Value"  
#HR.BindS  
  
# Salary Increase worked Ranges:   
HR\_linear$SalaryIncreaseLevel[HR\_linear$PercentSalaryHike >= Percentile\_00 & HR\_linear$PercentSalaryHike < Percentile\_33] = "Low\_Increase"  
HR\_linear$SalaryIncreaseLevel[HR\_linear$PercentSalaryHike >= Percentile\_33 & HR\_linear$PercentSalaryHike < Percentile\_67] = "Avg\_Increase"  
HR\_linear$SalaryIncreaseLevel[HR\_linear$PercentSalaryHike >= Percentile\_67 & HR\_linear$PercentSalaryHike <= Percentile\_100] = "High\_Increase"  
  
# Categoric Working Years  
  
# Working Years Percentiles  
Percentile\_00 = min(HR\_linear$TotalWorkingYears)  
Percentile\_33 = quantile(HR\_linear$TotalWorkingYears, 0.33333)  
Percentile\_67 = quantile(HR\_linear$TotalWorkingYears, 0.66667)  
Percentile\_100 = max(HR\_linear$TotalWorkingYears)  
  
# Values  
HR.BindW = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.BindW)[[2]] = "Value"  
#HR.BindW  
  
# Working Years Ranges:   
HR\_linear$WorkingYears[HR\_linear$TotalWorkingYears >= Percentile\_00 & HR\_linear$TotalWorkingYears < Percentile\_33] = "Lower\_Range"  
HR\_linear$WorkingYears[HR\_linear$TotalWorkingYears >= Percentile\_33 & HR\_linear$TotalWorkingYears < Percentile\_67] = "Mid\_Range"  
HR\_linear$WorkingYears[HR\_linear$TotalWorkingYears >= Percentile\_67 & HR\_linear$TotalWorkingYears <= Percentile\_100] = "Higher\_Range"  
  
# Categoric Years At Company  
  
# Years At Company Percentiles  
Percentile\_00 = min(HR\_linear$YearsAtCompany)  
Percentile\_33 = quantile(HR\_linear$YearsAtCompany, 0.33333)  
Percentile\_67 = quantile(HR\_linear$YearsAtCompany, 0.66667)  
Percentile\_100 = max(HR\_linear$YearsAtCompany)  
  
# Values  
HR.BindY = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.BindY)[[2]] = "Value"  
#HR.BindY  
  
# Years At Company Ranges:   
HR\_linear$CompanyYears[HR\_linear$YearsAtCompany >= Percentile\_00 & HR\_linear$YearsAtCompany < Percentile\_33] = "Lower\_Range"  
HR\_linear$CompanyYears[HR\_linear$YearsAtCompany >= Percentile\_33 & HR\_linear$YearsAtCompany < Percentile\_67] = "Mid\_Range"  
HR\_linear$CompanyYears[HR\_linear$YearsAtCompany >= Percentile\_67 & HR\_linear$YearsAtCompany <= Percentile\_100] = "Higher\_Range"  
  
# Categoric Years in Current Role  
  
# Years in Current Role Percentiles  
Percentile\_00 = min(HR\_linear$YearsInCurrentRole)  
Percentile\_33 = quantile(HR\_linear$YearsInCurrentRole, 0.33333)  
Percentile\_67 = quantile(HR\_linear$YearsInCurrentRole, 0.66667)  
Percentile\_100 = max(HR\_linear$YearsInCurrentRole)  
  
# Values  
HR.BindR = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.BindR)[[2]] = "Value"  
#HR.BindR  
  
# Years in Current Role Ranges:   
HR\_linear$RoleYear[HR\_linear$YearsInCurrentRole >= Percentile\_00 & HR\_linear$YearsInCurrentRole < Percentile\_33] = "Lower\_Range"  
HR\_linear$RoleYear[HR\_linear$YearsInCurrentRole >= Percentile\_33 & HR\_linear$YearsInCurrentRole < Percentile\_67] = "Mid\_Range"  
HR\_linear$RoleYear[HR\_linear$YearsInCurrentRole >= Percentile\_67 & HR\_linear$YearsInCurrentRole <= Percentile\_100] = "Higher\_Range"  
  
# Categoric Years No Promotion  
  
# Years No Promotion Percentiles  
Percentile\_00 = min(HR\_linear$YearsSinceLastPromotion)  
Percentile\_33 = quantile(HR\_linear$YearsSinceLastPromotion, 0.33333)  
Percentile\_67 = quantile(HR\_linear$YearsSinceLastPromotion, 0.66667)  
Percentile\_100 = max(HR\_linear$YearsSinceLastPromotion)  
  
# Values  
HR.BindP = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.BindP)[[2]] = "Value"  
#HR.BindP  
  
# Years No Promotion Ranges:   
HR\_linear$NoPromoYears[HR\_linear$YearsSinceLastPromotion >= Percentile\_00 & HR\_linear$YearsSinceLastPromotion < Percentile\_33] = "Lower\_Range"  
HR\_linear$NoPromoYears[HR\_linear$YearsSinceLastPromotion >= Percentile\_33 & HR\_linear$YearsSinceLastPromotion < Percentile\_67] = "Mid\_Range"  
HR\_linear$NoPromoYears[HR\_linear$YearsSinceLastPromotion >= Percentile\_67 & HR\_linear$YearsSinceLastPromotion <= Percentile\_100] = "Higher\_Range"  
  
# Categoric Years Current Manager  
  
# Years Current Manager Percentiles  
Percentile\_00 = min(HR\_linear$YearsWithCurrManager)  
Percentile\_33 = quantile(HR\_linear$YearsWithCurrManager, 0.33333)  
Percentile\_67 = quantile(HR\_linear$YearsWithCurrManager, 0.66667)  
Percentile\_100 = max(HR\_linear$YearsWithCurrManager)  
  
# Values  
HR.BindM = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.BindM)[[2]] = "Value"  
#HR.BindM  
  
# Years Current Manager Ranges:   
HR\_linear$ManagerYears[HR\_linear$YearsWithCurrManager >= Percentile\_00 & HR\_linear$YearsWithCurrManager < Percentile\_33] = "Lower\_Range"  
HR\_linear$ManagerYears[HR\_linear$YearsWithCurrManager >= Percentile\_33 & HR\_linear$YearsWithCurrManager < Percentile\_67] = "Mid\_Range"  
HR\_linear$ManagerYears[HR\_linear$YearsWithCurrManager >= Percentile\_67 & HR\_linear$YearsWithCurrManager <= Percentile\_100] = "Higher\_Range"  
  
# Remove Numerical values categorized  
HR\_linear<-HR\_linear[c(-1,-2,-5,-7,-12,-18,-19,-20,-22,-26,-29,-30,-31,-32)]  
  
# Convert all other Numerical values to factors  
HR\_linear<-lapply(HR\_linear, function(x){as.factor(x)})  
HR\_linear = as.data.frame(HR\_linear)  
str(HR\_linear)

[structure removed for ease of reading]

#summary(HR\_linear)  
  
Percentiles.HR<-cbind(HR.BindA,HR.BindH,HR.BindDR,HR.BindMR,HR.BindI,HR.BindD,HR.BindC,HR.BindS,HR.BindW,HR.BindY,HR.BindR,HR.BindP,HR.BindM)  
colnames(Percentiles.HR)<-c("Age","HourlyRate","DailyRate","MonthlyRate","MonthlyIncome","HomeDistance","CompaniesWorked","SalaryIncrease","WorkingYears","YearsAtCompany","YearsInRole","NoPromoYears","YearsWManager")  
if("knitr" %in% rownames(installed.packages()) == FALSE) {install.packages('knitr') }  
library(knitr)

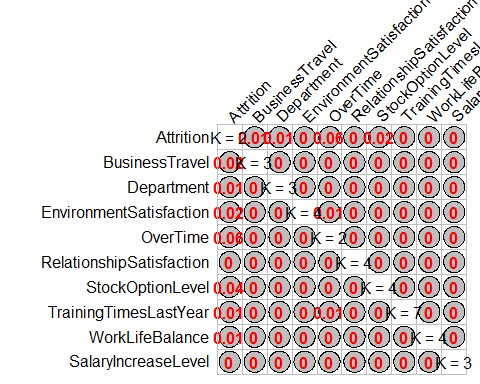
kable(t(Percentiles.HR),digits=0, format="markdown", padding =2, format.args = list(big.mark = ","))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Percentile\_00 | Percentile\_33 | Percentile\_67 | Percentile\_100 |
| Age | 18 | 32 | 40 | 60 |
| HourlyRate | 30 | 54 | 78 | 100 |
| DailyRate | 102 | 573 | 1,039 | 1,499 |
| MonthlyRate | 2,094 | 10,035 | 18,615 | 26,999 |
| MonthlyIncome | 1,009 | 3,632 | 6,529 | 19,999 |
| HomeDistance | 1 | 3 | 10 | 29 |
| CompaniesWorked | 0 | 1 | 3 | 9 |
| SalaryIncrease | 11 | 13 | 16 | 25 |
| WorkingYears | 0 | 7 | 12 | 40 |
| YearsAtCompany | 0 | 4 | 8 | 40 |
| YearsInRole | 0 | 2 | 6 | 18 |
| NoPromoYears | 0 | 0 | 2 | 15 |
| YearsWManager | 0 | 2 | 6 | 17 |

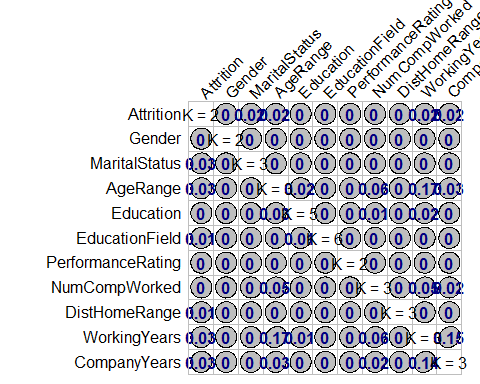
grid.arrange(tableGrob(t(format(Percentiles.HR,digits=0,big.mark=",")),   
 theme=ttheme\_default(core=list(fg\_params=list(fontface=3),big.mark = ","),  
 colhead=list(fg\_params=list(col="navyblue", fontface=4L)), rowhead=list(fg\_params=list(col="navyblue", fontface=3L)))))

[table removed for ease of reading]

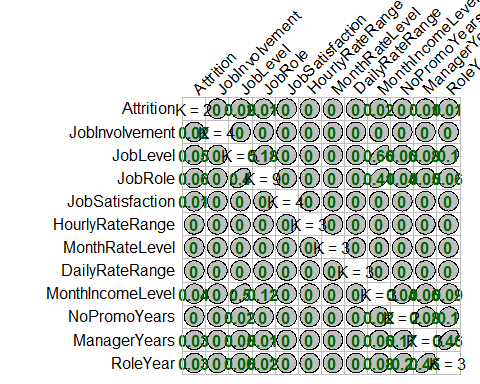
varCompany.set<- c("Attrition","BusinessTravel","Department","EnvironmentSatisfaction","OverTime","RelationshipSatisfaction","StockOptionLevel","TrainingTimesLastYear", "WorkLifeBalance", "SalaryIncreaseLevel")  
varPerson.set<- c("Attrition","Gender","MaritalStatus","AgeRange","Education","EducationField","PerformanceRating", "NumCompWorked","DistHomeRange","WorkingYears","CompanyYears")  
varJob.set<- c("Attrition","JobInvolvement","JobLevel","JobRole","JobSatisfaction","HourlyRateRange","MonthRateLevel","DailyRateRange", "MonthIncomeLevel", "NoPromoYears","ManagerYears","RoleYear")  
  
Frame1<- subset(HR\_linear, select = varCompany.set)  
Frame2<- subset(HR\_linear, select = varPerson.set)  
Frame3<- subset(HR\_linear, select = varJob.set)  
  
GKmatrix1<- GKtauDataframe(Frame1)  
plot(GKmatrix1, corrColors = "red")



GKmatrix1<- GKtauDataframe(Frame2)  
plot(GKmatrix1, corrColors = "navyblue")



GKmatrix1<- GKtauDataframe(Frame3)  
plot(GKmatrix1, corrColors = "darkgreen")



#Logistic Regression Model  
Attrition.Model<-glm(Attrition~.,data=HR\_linear, family = binomial())  
summary(Attrition.Model)

##   
## Call:  
## glm(formula = Attrition ~ ., family = binomial(), data = HR\_linear)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.8221 -0.4239 -0.1935 -0.0608 3.4447   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -10.75546 591.97139 -0.018 0.985504   
## BusinessTravelTravel\_Frequently 1.97433 0.44861 4.401 1.08e-05 \*\*\*  
## BusinessTravelTravel\_Rarely 0.91970 0.41169 2.234 0.025485 \*   
## DepartmentResearch & Development 14.56879 591.97061 0.025 0.980366   
## DepartmentSales 13.64583 591.97079 0.023 0.981609   
## Education2 0.13897 0.35400 0.393 0.694637   
## Education3 0.15916 0.31322 0.508 0.611349   
## Education4 0.21933 0.34823 0.630 0.528791   
## Education5 0.13534 0.66229 0.204 0.838085   
## EducationFieldLife Sciences -1.21045 0.90971 -1.331 0.183325   
## EducationFieldMarketing -0.54666 0.96623 -0.566 0.571551   
## EducationFieldMedical -1.15912 0.90834 -1.276 0.201927   
## EducationFieldOther -1.05483 0.97815 -1.078 0.280856   
## EducationFieldTechnical Degree -0.07002 0.92176 -0.076 0.939451   
## EnvironmentSatisfaction2 -1.11827 0.29978 -3.730 0.000191 \*\*\*  
## EnvironmentSatisfaction3 -1.20927 0.27365 -4.419 9.91e-06 \*\*\*  
## EnvironmentSatisfaction4 -1.53958 0.28182 -5.463 4.68e-08 \*\*\*  
## GenderMale 0.47825 0.20025 2.388 0.016925 \*   
## JobInvolvement2 -1.48650 0.39191 -3.793 0.000149 \*\*\*  
## JobInvolvement3 -1.74282 0.36722 -4.746 2.08e-06 \*\*\*  
## JobInvolvement4 -2.52586 0.51498 -4.905 9.35e-07 \*\*\*  
## JobLevel2 -1.54845 0.52632 -2.942 0.003261 \*\*   
## JobLevel3 -0.63142 0.71967 -0.877 0.380283   
## JobLevel4 -1.63014 0.99342 -1.641 0.100810   
## JobLevel5 0.72121 1.25275 0.576 0.564817   
## JobRoleHuman Resources 15.00953 591.97064 0.025 0.979772   
## JobRoleLaboratory Technician 0.76607 0.63039 1.215 0.224276   
## JobRoleManager -0.52825 1.05965 -0.499 0.618120   
## JobRoleManufacturing Director 0.36403 0.57706 0.631 0.528151   
## JobRoleResearch Director -2.14713 1.10191 -1.949 0.051349 .   
## JobRoleResearch Scientist -0.52874 0.65182 -0.811 0.417269   
## JobRoleSales Executive 2.26362 1.23758 1.829 0.067388 .   
## JobRoleSales Representative 2.03437 1.33341 1.526 0.127086   
## JobSatisfaction2 -0.63516 0.29573 -2.148 0.031733 \*   
## JobSatisfaction3 -0.67078 0.26230 -2.557 0.010549 \*   
## JobSatisfaction4 -1.32957 0.27730 -4.795 1.63e-06 \*\*\*  
## MaritalStatusMarried 0.37415 0.29782 1.256 0.209006   
## MaritalStatusSingle 0.86860 0.43003 2.020 0.043397 \*   
## OverTimeYes 2.18386 0.21598 10.111 < 2e-16 \*\*\*  
## PerformanceRating4 -0.14814 0.33200 -0.446 0.655450   
## RelationshipSatisfaction2 -0.77408 0.30556 -2.533 0.011300 \*   
## RelationshipSatisfaction3 -0.95383 0.27403 -3.481 0.000500 \*\*\*  
## RelationshipSatisfaction4 -0.90046 0.27236 -3.306 0.000946 \*\*\*  
## StockOptionLevel1 -1.02983 0.33514 -3.073 0.002121 \*\*   
## StockOptionLevel2 -0.89991 0.47163 -1.908 0.056380 .   
## StockOptionLevel3 -0.09687 0.49895 -0.194 0.846057   
## TrainingTimesLastYear1 -1.21408 0.61198 -1.984 0.047271 \*   
## TrainingTimesLastYear2 -1.29649 0.45547 -2.846 0.004420 \*\*   
## TrainingTimesLastYear3 -1.43809 0.46161 -3.115 0.001837 \*\*   
## TrainingTimesLastYear4 -1.18525 0.52739 -2.247 0.024614 \*   
## TrainingTimesLastYear5 -1.76607 0.57347 -3.080 0.002073 \*\*   
## TrainingTimesLastYear6 -2.13607 0.68857 -3.102 0.001921 \*\*   
## WorkLifeBalance2 -1.23098 0.40292 -3.055 0.002249 \*\*   
## WorkLifeBalance3 -1.73684 0.37493 -4.632 3.61e-06 \*\*\*  
## WorkLifeBalance4 -1.14898 0.44940 -2.557 0.010568 \*   
## AgeRangeLower\_Range 0.73932 0.30033 2.462 0.013829 \*   
## AgeRangeMid\_Range 0.04254 0.27367 0.155 0.876474   
## HourlyRateRangeLow\_Range -0.17638 0.24021 -0.734 0.462767   
## HourlyRateRangeMid\_Range -0.20467 0.23583 -0.868 0.385476   
## DailyRateRangeLow\_Range 0.55288 0.24346 2.271 0.023152 \*   
## DailyRateRangeMid\_Range 0.47637 0.24559 1.940 0.052416 .   
## MonthRateLevelLow\_Income -0.21245 0.24084 -0.882 0.377727   
## MonthRateLevelMid\_Income 0.07881 0.23373 0.337 0.735978   
## MonthIncomeLevelLow\_Income 0.29736 0.58747 0.506 0.612736   
## MonthIncomeLevelMid\_Income -0.18229 0.46001 -0.396 0.691911   
## DistHomeRangeLow\_Distance -1.07635 0.26194 -4.109 3.97e-05 \*\*\*  
## DistHomeRangeMid\_Distance -0.66039 0.22723 -2.906 0.003658 \*\*   
## NumCompWorkedLow\_Number -1.19190 0.35831 -3.326 0.000879 \*\*\*  
## NumCompWorkedMid\_Number -0.64634 0.25362 -2.548 0.010819 \*   
## SalaryIncreaseLevelHigh\_Increase 0.33967 0.26592 1.277 0.201484   
## SalaryIncreaseLevelLow\_Increase 0.49711 0.24994 1.989 0.046707 \*   
## WorkingYearsLower\_Range 0.65099 0.42321 1.538 0.123995   
## WorkingYearsMid\_Range 0.46081 0.30686 1.502 0.133173   
## CompanyYearsLower\_Range -0.17100 0.46833 -0.365 0.715017   
## CompanyYearsMid\_Range -0.08981 0.41168 -0.218 0.827306   
## RoleYearLower\_Range 0.83311 0.43739 1.905 0.056817 .   
## RoleYearMid\_Range 0.27031 0.37770 0.716 0.474182   
## NoPromoYearsMid\_Range -0.71065 0.24977 -2.845 0.004439 \*\*   
## ManagerYearsLower\_Range 0.37542 0.41858 0.897 0.369776   
## ManagerYearsMid\_Range -0.51274 0.40720 -1.259 0.207968   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1298.58 on 1469 degrees of freedom  
## Residual deviance: 764.83 on 1390 degrees of freedom  
## AIC: 924.83  
##   
## Number of Fisher Scoring iterations: 15

plot(Attrition.Model)

|  |  |
| --- | --- |
|  |  |
|  |  |

coef(Attrition.Model)

[coef removed for ease of reading]

Linear regression of categorical data doesn’t show high associations to attrition, with the hisghest one being Attrition and Overtime at 0.06.

Considering associations of other variables the highest association were:

Age Range to Working Years Working Years to Years in Company Job level and job role to Monthly Income level and, Time in a role to time with a manager

### Models

#### Association Rule Mining

References:

• https://rcompanion.org/handbook/E\_05.html

• https://towardsdatascience.com/association-rule-mining-in-r-ddf2d044ae50

• https://www.datacamp.com/community/tutorials/market-basket-analysis-r  
  
library(arules)  
library(arulesViz)  
library(RColorBrewer)  
library(gridExtra)  
library(grid)  
library(ggplot2)  
library(lattice)  
  
#Data Assessment  
HR\_arm <- HR\_clean  
str(HR\_arm)

[structure removed for ease of reading]

Actions required: 1. Eliminate Employee ID 2. Redundant Attributes removed Daily Rate Hourly Rate Monthly Rate 3. Other Numerical Values need to be converted to Factors

# Data Transformation  
  
# Remove Redundant, None added value attributes  
HR\_arm<-HR\_arm[c(-1,-5,-12,-19)]  
  
#Create a Categoric Income Label based on Percentiles  
  
 # Determining percentiles  
Percentile\_00 = min(HR\_arm$MonthlyIncome)  
Percentile\_33 = quantile(HR\_arm$MonthlyIncome, 0.33333)  
Percentile\_67 = quantile(HR\_arm$MonthlyIncome, 0.66667)  
Percentile\_100 = max(HR\_arm$MonthlyIncome)  
  
 # Values  
HR.Bind = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.Bind)[[2]] = "Value"  
HR.Bind

## Value  
## Percentile\_00 1009.000  
## Percentile\_33 3631.647  
## Percentile\_67 6528.735  
## Percentile\_100 19999.000

# Grouping  
HR\_arm$Group[HR\_arm$MonthlyIncome >= Percentile\_00 & HR\_arm$MonthlyIncome < Percentile\_33] = "Low\_Income"  
HR\_arm$Group[HR\_arm$MonthlyIncome >= Percentile\_33 & HR\_arm$MonthlyIncome < Percentile\_67] = "Mid\_Income"  
HR\_arm$Group[HR\_arm$MonthlyIncome >= Percentile\_67 & HR\_arm$MonthlyIncome <= Percentile\_100] = "High\_Income"  
  
 # Remove Numerical "values"Monthly Income"  
HR\_arm<-HR\_arm[-15]  
  
 # Convert all other Numerical values to factors  
HR\_arm<-lapply(HR\_arm, function(x){as.factor(x)})  
HR\_arm = as.data.frame(HR\_arm)  
str(HR\_arm)

[structure removed for ease of reading]

# Convert to Transactional Data  
HR\_Trans = as(HR\_arm, "transactions")  
HR\_Trans

## transactions in sparse format with  
## 1470 transactions (rows) and  
## 303 items (columns)

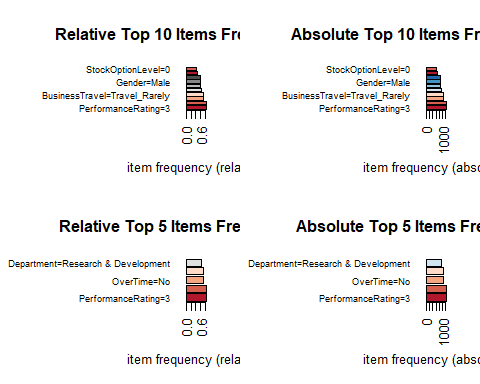
Data set as transactions! Lets take a look

# Information about the transactions data  
  
summary(HR\_Trans)

[summary removed for ease of reading]

* 1470 rows (elements/itemsets/transactions) and 303 columns (items)
* Dense Matrix based on Percent of Non-Zero cells in the matrix (Density) of 0.09240924.
* Most Frequency Items:
  + PerformanceRating=3: 1244
  + Attrition=No: 1233
  + OverTime=No: 1054
  + BusinessTravel=Travel\_Rarely: 1043
  + Department=Research & Development: 961

par(mfrow=c(2,2))  
  
# Item Frequency Plot Top 10 Relative  
arules::itemFrequencyPlot(HR\_Trans,support = 0.2, cex.names=0.7, topN=10, col=brewer.pal(8,'RdGy'), type="relative",main="Relative Top 10 Items Frequency Plot", horiz=TRUE)  
  
# Item Frequency Plot Top 10 Absolute  
itemFrequencyPlot(HR\_Trans,support = 0.2, cex.names=0.7, topN=10, col=brewer.pal(8,'RdBu'), type="absolute", main="Absolute Top 10 Items Frequency Plot",horiz=TRUE)  
  
# Item Frequency Plot for top 5 Relative  
itemFrequencyPlot(HR\_Trans,support = 0.2, cex.names=0.7, topN=5, col=brewer.pal(8,'RdGy'),type="relative", main="Relative Top 5 Items Frequency Plot", horiz=TRUE)  
  
# Item Frequency Plot for top 5 most frequent items  
itemFrequencyPlot(HR\_Trans,support = 0.2, cex.names=0.7, topN= 5,col=brewer.pal(8,'RdBu'), type="absolute", main="Absolute Top 5 Items Frequency Plot",horiz=TRUE)

 “Attrition= No” is in the top of the list along with No Overtime, Travel Rarely and Performance Rating =3

# Apriori Rules with Support = 0.1 and Confidence 0.5  
HR\_Rules1<-apriori(HR\_Trans,parameter = list(support=0.1, confidence =0.5, maxlen = 305))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.5 0.1 1 none FALSE TRUE 5 0.1 1  
## maxlen target ext  
## 305 rules FALSE  
##   
## Absolute minimum support count: 147   
##

[partial results removed for ease of reading]

HR\_Rules1

## set of 10478 rules

## Changing some parameters  
 ### For stronger rules: Increased confidence.  
 ### For lenghtier rules increase the maxlen parameter.  
 ### To eliminate shorter rules decrease the minlen parameter.  
  
# Apriori Rules with Support = 0.1 and Confidence 0.9 max items 30 min items 3  
HR\_Rules2<-apriori(HR\_Trans,parameter = list(support=0.1, confidence =0.9, maxlen = 30, minlen = 3))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.9 0.1 1 none FALSE TRUE 5 0.1 3  
## maxlen target ext  
## 30 rules FALSE  
##   
## Absolute minimum support count: 147   
##

[partial results removed for ease of reading]

HR\_Rules2

## set of 921 rules

# Apriori Rules with Support = 0.01 and Confidence 0.8 and RHS fixed to Attrition =Yes  
HR\_Rules3<-apriori(HR\_Trans,parameter = list(support=0.01, confidence =0.8, maxlen = 30), appearance = list(rhs="Attrition=Yes"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.8 0.1 1 none FALSE TRUE 5 0.01 1  
## maxlen target ext  
## 30 rules FALSE  
##   
## Absolute minimum support count: 14   
##

[partial results removed for ease of reading]

HR\_Rules3

## set of 243 rules

# Apriori Rules with Support = 0.1 and Confidence 0.9 and RHS fixed to Attrition =No  
HR\_Rules4<-apriori(HR\_Trans,parameter = list(support=0.1, confidence =0.8, maxlen = 30), appearance = list(rhs="Attrition=No"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.8 0.1 1 none FALSE TRUE 5 0.1 1  
## maxlen target ext  
## 30 rules FALSE  
##   
## Absolute minimum support count: 147   
##

[partial results removed for ease of reading]

HR\_Rules4

## set of 1557 rules

Based on 303 items and 1,470 transactions and changing parameters created rules:

* First set of rules, created 10,478 rules
* Second set of rules created 921 rules
* Third set of rules (fixing the RHS to Attrition=No) 243 rules
* Fourth set of rules (fixing the RHS to Attrition=Yes) 1557 rules

Support is an indication of how frequently the itemset appears in the dataset. For Attrition = Yes support was reduced to 0.01 as opposed to the other models that considered support = 0.1

Confidence is an indication of how often the rule has been found to be true. All rules generated with confidence >= 80%

#Rules Summaries (just for Rules with Attrition Fixed)  
  
# Attrition = Yes  
  
summary(HR\_Rules3)

[summary removed for ease of reading]

# Attrition = No  
  
summary(HR\_Rules4)

[partial results removed for ease of reading]

For Attrition = Yes

* Parameter Specification: Support= 0.01 and Confidence = 0.8
* A length of 6 items has the most rules (100) while a length of 10 items has only one \* Summary of Quality Measures: Min and Max Values for Support, Confidence and Lift shown For Attrition = No
* Parameter Specification: Support= 0.1 and Confidence = 0.8 (Same confidence much lower Support than the prior one)
* A length of 4 items has the most rules (688) while a length of 1 item has only one
* Summary of Quality Measures: Min and Max Values for Support, Confidence and Lift shown

Next: Looking at the top 20 rules considering 1 set of rules created without a fix RHS and the 2 RHS fixed rules

# Top 100 Rules for second set of Rules (Not Fixed)  
  
inspect(head(sort(HR\_Rules2, by = "confidence"), 100))

[rules removed for ease of reading]

# Top 20 Rules for rules with RHS at Attrition = Yes  
  
inspect(head(sort(HR\_Rules3, by = "confidence"), 20))

## lhs rhs support confidence lift count  
## [1] {MaritalStatus=Single,   
## OverTime=Yes,   
## YearsWithCurrManager=0,   
## Group=Low\_Income} => {Attrition=Yes} 0.01088435 1.0000000 6.202532 16  
## [2] {JobLevel=1,   
## MaritalStatus=Single,   
## OverTime=Yes,   
## YearsWithCurrManager=0} => {Attrition=Yes} 0.01156463 1.0000000 6.202532 17  
## [3] {MaritalStatus=Single,   
## OverTime=Yes,   
## YearsInCurrentRole=0,   
## YearsWithCurrManager=0,   
## Group=Low\_Income} => {Attrition=Yes} 0.01020408 1.0000000 6.202532 15  
## [4] {JobLevel=1,   
## MaritalStatus=Single,   
## OverTime=Yes,   
## YearsInCurrentRole=0,   
## YearsWithCurrManager=0} => {Attrition=Yes} 0.01088435 1.0000000 6.202532 16  
## [5] {JobLevel=1,   
## OverTime=Yes,   
## StockOptionLevel=0,   
## YearsInCurrentRole=0,   
## YearsWithCurrManager=0} => {Attrition=Yes} 0.01156463 1.0000000 6.202532 17  
## [6] {JobLevel=1,   
## MaritalStatus=Single,   
## OverTime=Yes,   
## YearsWithCurrManager=0,   
## Group=Low\_Income} => {Attrition=Yes} 0.01088435 1.0000000 6.202532 16  
## [7] {MaritalStatus=Single,   
## OverTime=Yes,   
## StockOptionLevel=0,   
## YearsWithCurrManager=0,   
## Group=Low\_Income} => {Attrition=Yes} 0.01088435 1.0000000 6.202532 16  
## [8] {JobLevel=1,   
## MaritalStatus=Single,   
## OverTime=Yes,   
## StockOptionLevel=0,   
## YearsWithCurrManager=0} => {Attrition=Yes} 0.01156463 1.0000000 6.202532 17  
## [9] {JobLevel=1,   
## OverTime=Yes,   
## StockOptionLevel=0,   
## YearsSinceLastPromotion=0,   
## YearsWithCurrManager=0} => {Attrition=Yes} 0.01088435 1.0000000 6.202532 16  
## [10] {BusinessTravel=Travel\_Frequently,   
## JobLevel=1,   
## PerformanceRating=3,   
## YearsInCurrentRole=0,   
## YearsWithCurrManager=0,   
## Group=Low\_Income} => {Attrition=Yes} 0.01088435 1.0000000 6.202532 16  
## [11] {JobLevel=1,   
## MaritalStatus=Single,   
## OverTime=Yes,   
## YearsInCurrentRole=0,   
## YearsWithCurrManager=0,   
## Group=Low\_Income} => {Attrition=Yes} 0.01020408 1.0000000 6.202532 15  
## [12] {MaritalStatus=Single,   
## OverTime=Yes,   
## StockOptionLevel=0,   
## YearsInCurrentRole=0,   
## YearsWithCurrManager=0,   
## Group=Low\_Income} => {Attrition=Yes} 0.01020408 1.0000000 6.202532 15  
## [13] {JobLevel=1,   
## MaritalStatus=Single,   
## OverTime=Yes,   
## StockOptionLevel=0,   
## YearsInCurrentRole=0,   
## YearsWithCurrManager=0} => {Attrition=Yes} 0.01088435 1.0000000 6.202532 16  
## [14] {JobLevel=1,   
## OverTime=Yes,   
## StockOptionLevel=0,   
## YearsInCurrentRole=0,   
## YearsWithCurrManager=0,   
## Group=Low\_Income} => {Attrition=Yes} 0.01088435 1.0000000 6.202532 16  
## [15] {JobLevel=1,   
## MaritalStatus=Single,   
## OverTime=Yes,   
## StockOptionLevel=0,   
## YearsWithCurrManager=0,   
## Group=Low\_Income} => {Attrition=Yes} 0.01088435 1.0000000 6.202532 16  
## [16] {JobLevel=1,   
## OverTime=Yes,   
## StockOptionLevel=0,   
## YearsSinceLastPromotion=0,   
## YearsWithCurrManager=0,   
## Group=Low\_Income} => {Attrition=Yes} 0.01020408 1.0000000 6.202532 15  
## [17] {JobLevel=1,   
## MaritalStatus=Single,   
## OverTime=Yes,   
## StockOptionLevel=0,   
## YearsInCurrentRole=0,   
## YearsWithCurrManager=0,   
## Group=Low\_Income} => {Attrition=Yes} 0.01020408 1.0000000 6.202532 15  
## [18] {JobLevel=1,   
## OverTime=Yes,   
## StockOptionLevel=0,   
## YearsInCurrentRole=0} => {Attrition=Yes} 0.01292517 0.9500000 5.892405 19  
## [19] {JobLevel=1,   
## OverTime=Yes,   
## StockOptionLevel=0,   
## YearsWithCurrManager=0} => {Attrition=Yes} 0.01292517 0.9500000 5.892405 19  
## [20] {JobLevel=1,   
## MaritalStatus=Single,   
## OverTime=Yes,   
## YearsInCurrentRole=0} => {Attrition=Yes} 0.01224490 0.9473684 5.876083 18

# Top 20 Rules for rules with RHS at Attrition = No  
  
inspect(head(sort(HR\_Rules4, by = "confidence"), 20))

## lhs rhs support confidence lift count  
## [1] {Department=Research & Development,   
## OverTime=No,   
## StockOptionLevel=1,   
## WorkLifeBalance=3} => {Attrition=No} 0.1081633 0.9754601 1.162957 159  
## [2] {BusinessTravel=Travel\_Rarely,   
## Department=Research & Development,   
## OverTime=No,   
## Group=High\_Income} => {Attrition=No} 0.1006803 0.9673203 1.153253 148  
## [3] {OverTime=No,   
## StockOptionLevel=1,   
## WorkLifeBalance=3} => {Attrition=No} 0.1680272 0.9648438 1.150300 247  
## [4] {EnvironmentSatisfaction=4,   
## OverTime=No,   
## WorkLifeBalance=3} => {Attrition=No} 0.1224490 0.9625668 1.147586 180  
## [5] {Department=Research & Development,   
## OverTime=No,   
## YearsWithCurrManager=2} => {Attrition=No} 0.1034014 0.9620253 1.146940 152  
## [6] {JobLevel=2,   
## StockOptionLevel=1,   
## Group=Mid\_Income} => {Attrition=No} 0.1034014 0.9620253 1.146940 152  
## [7] {BusinessTravel=Travel\_Rarely,   
## JobLevel=2,   
## OverTime=No,   
## WorkLifeBalance=3} => {Attrition=No} 0.1034014 0.9620253 1.146940 152  
## [8] {EnvironmentSatisfaction=4,   
## OverTime=No,   
## PerformanceRating=3,   
## WorkLifeBalance=3} => {Attrition=No} 0.1027211 0.9617834 1.146652 151  
## [9] {EducationField=Life Sciences,   
## OverTime=No,   
## StockOptionLevel=1} => {Attrition=No} 0.1190476 0.9615385 1.146360 175  
## [10] {JobLevel=2,   
## OverTime=No,   
## StockOptionLevel=1} => {Attrition=No} 0.1000000 0.9607843 1.145461 147  
## [11] {Department=Research & Development,   
## MaritalStatus=Married,   
## OverTime=No,   
## WorkLifeBalance=3} => {Attrition=No} 0.1156463 0.9604520 1.145064 170  
## [12] {Department=Research & Development,   
## JobLevel=2,   
## WorkLifeBalance=3} => {Attrition=No} 0.1142857 0.9600000 1.144526 168  
## [13] {Department=Research & Development,   
## JobLevel=2,   
## OverTime=No,   
## PerformanceRating=3} => {Attrition=No} 0.1136054 0.9597701 1.144251 167  
## [14] {Department=Research & Development,   
## Gender=Female,   
## OverTime=No,   
## WorkLifeBalance=3} => {Attrition=No} 0.1074830 0.9575758 1.141635 158  
## [15] {OverTime=No,   
## PerformanceRating=3,   
## StockOptionLevel=1,   
## WorkLifeBalance=3} => {Attrition=No} 0.1380952 0.9575472 1.141601 203  
## [16] {JobSatisfaction=4,   
## OverTime=No,   
## WorkLifeBalance=3} => {Attrition=No} 0.1210884 0.9569892 1.140936 178  
## [17] {Department=Research & Development,   
## TrainingTimesLastYear=3,   
## WorkLifeBalance=3} => {Attrition=No} 0.1176871 0.9558011 1.139520 173  
## [18] {MaritalStatus=Married,   
## OverTime=No,   
## StockOptionLevel=1,   
## WorkLifeBalance=3} => {Attrition=No} 0.1176871 0.9558011 1.139520 173  
## [19] {Department=Research & Development,   
## JobLevel=2,   
## PerformanceRating=3} => {Attrition=No} 0.1551020 0.9539749 1.137342 228  
## [20] {BusinessTravel=Travel\_Rarely,   
## Department=Research & Development,   
## Group=High\_Income} => {Attrition=No} 0.1360544 0.9523810 1.135442 200

The first set of rules provides insight in regards to performance rating, department information, stock option level and job level but no information about attrition. By fixing the RHS to Attrition = Yes and Attrition = No rules provide more insight.

With Attrition = Yes, the most frequent factors in the top 20 rules are:

* Marital Status = Single. In 13 out of the 20 rules
* Overtime = Yes. In 18 out of the 20 rules
* Years with current Manager = 0. In 16 out of the 20 rules
* Years in current Role = 0. In 12 out of the 20 rules
* Low Income. In 10 out of the 20 rules

With Attrition = No, the most frequent factors in the top 20 rules are:

* Department=Research & Development. In 10 out of the 20 rules
* OverTime=No. In 15 out of the 20 rules
* StockOptionLevel=1. In 6 out of the 20 rules
* WorkLifeBalance=3. In 11 out of the 20 rules

### Rules with Confidence > 40 and 50%  
   
## Attrition = Yes  
subsetRulesYes<-HR\_Rules3[quality(HR\_Rules3)$confidence>0.4]  
   
## Attrition = No  
subsetRulesNo<-HR\_Rules4[quality(HR\_Rules4)$confidence>0.5]  
  
### Plots  
  
## Scatter  
  
# Attrition = Yes  
plot(subsetRulesYes)

## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.

# Attrition = No  
plot(subsetRulesNo)

## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.

|  |  |
| --- | --- |
|  |  |

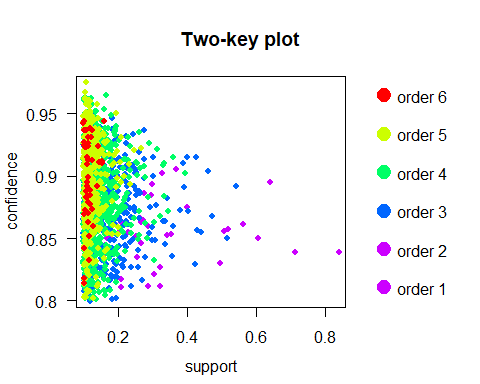
## Two-Key   
  
# Attrition = Yes  
plot(subsetRulesYes, method = "two-key plot")

## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.

[results removed for brevity]

# Attrition = No  
plot(subsetRulesNo, method = "two-key plot")

## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.



## Matrix 3D  
  
# Attrition = Yes  
plot(subsetRulesYes, method = "matrix3d")

## Warning in plot.rules(subsetRulesYes, method = "matrix3d"): method 'matrix3D' is  
## deprecated use method 'matrix' with engine '3d'

## Itemsets in Antecedent (LHS)  
## NULL  
## Itemsets in Consequent (RHS)  
## NULL

# Attrition = No  
plot(subsetRulesNo, method = "matrix3d")

## Warning in plot.rules(subsetRulesNo, method = "matrix3d"): method 'matrix3D' is  
## deprecated use method 'matrix' with engine '3d'

## Itemsets in Antecedent (LHS)  
## NULL  
## Itemsets in Consequent (RHS)  
## NULL

|  |  |
| --- | --- |
|  |  |

### Interactive Scatter-Plot  
  
# Attrition = Yes  
plotly\_arules(subsetRulesYes)

# Attrition = No  
plotly\_arules(subsetRulesNo)

#### Graph Based Visualizations  
  
### Subrules  
### Selecting 20 Rules with the Highest Confidence for each set  
  
 ## Attrition = Yes  
top20.subRulesYes<-head(subsetRulesYes, n = 20, by ="confidence")  
  
 ## Attrition = No  
top20.subRulesNo<-head(subsetRulesNo, n = 20, by ="confidence")  
  
### 20 Rules Plots  
  
 ## Attrition = Yes  
plot(top20.subRulesYes, method = "graph", engine = "htmlwidget")

## Attrition = No  
plot(top20.subRulesNo, method = "graph", engine = "htmlwidget")

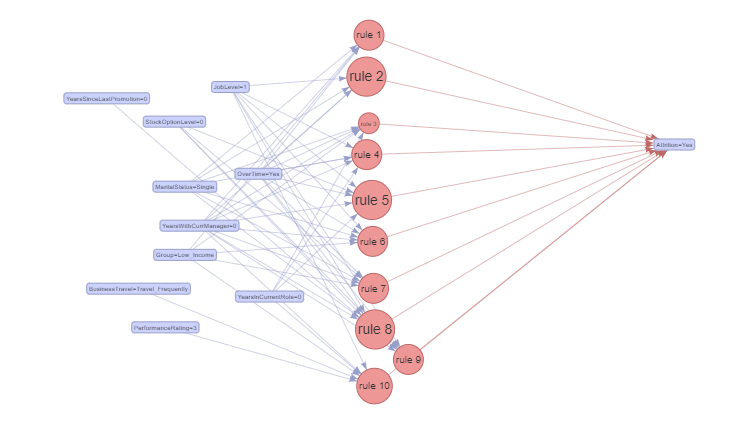
### Selecting 10 Rules with the Highest Confidence for each set  
  
 ## Attrition = Yes  
top10.subRulesYes<-head(subsetRulesYes, n = 10, by ="confidence")  
  
 ## Attrition = No  
top10.subRulesNo<-head(subsetRulesNo, n = 10, by ="confidence")  
  
### 10 Rules Plots  
  
 ## Attrition = Yes  
plot(top10.subRulesYes, method = "graph", engine = "htmlwidget")

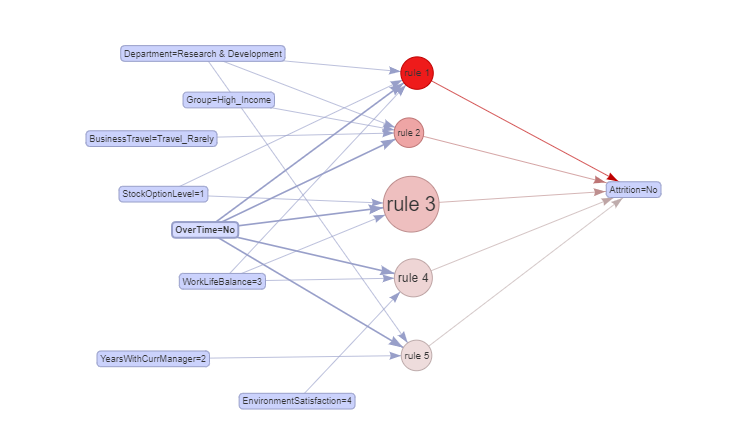
## Attrition = No  
plot(top10.subRulesNo, method = "graph", engine = "htmlwidget")

### Selecting 5 Rules with the Highest Confidence for each set  
  
 ## Attrition = Yes  
top5.subRulesYes<-head(subsetRulesYes, n = 5, by ="confidence")  
  
 ## Attrition = No  
top5.subRulesNo<-head(subsetRulesNo, n = 5, by ="confidence")  
  
## 5 Rules Plots  
  
 ## Attrition = Yes  
plot(top5.subRulesYes, method = "graph", engine = "htmlwidget")

## Attrition = No  
plot(top5.subRulesNo, method = "graph", engine = "htmlwidget")

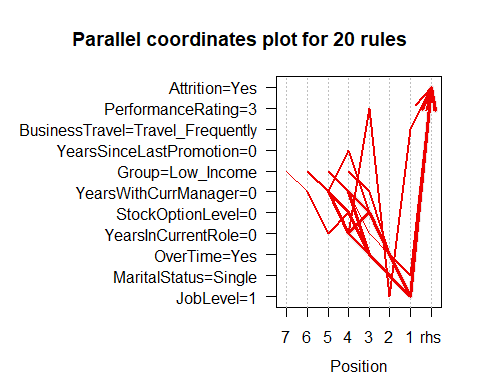
[partial results removed for ease of reading]



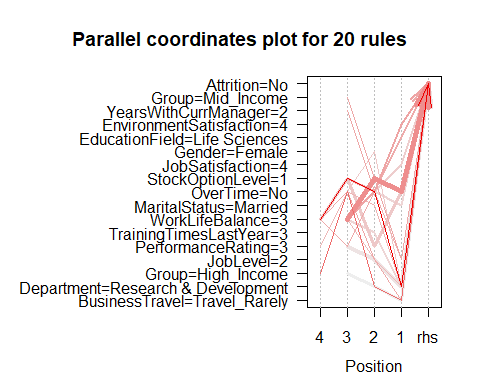


Graphs 1 and 2: Rules with high lift have low support Graphs 3 and 4: Rules with High confidence and low support have around 7 or 8 items. High support 5 or 6 items

### Selecting 20 Rules with the Highest Lift  
  
 ## Attrition = Yes  
top20.subRulesYesL<-head(subsetRulesYes, n = 20, by ="lift")  
  
 ## Attrition = No  
top20.subRulesNoL<-head(subsetRulesNo, n = 20, by ="lift")  
  
### 20 Rules Plots  
  
 ## Attrition = Yes  
plot(top20.subRulesYesL, method = "paracoord")



## Attrition = No  
plot(top20.subRulesNoL, method = "paracoord")



#### K-means Clustering

K-means clustering is used to visualize patterns in how the attributes contribute to the creation of groups of employees.

xc <- HR\_clean  
x\_factors <- Filter(is.factor, xc)  
head(x\_factors)

[results removed for ease of reading]

## Kmeans needs a matrix/dataframe of all numbers  
# remove employee number and attrition yes/no to start with  
xc <-HR\_clean  
xc\_att <-HR\_clean  
xc\_att <- xc[,c(2:32)] # keep a version of the data with attrition so we can compare the impact of attrition on groups  
xc <- xc[,c(2,4:32)]  
xc[] <- lapply(xc, function(x) as.numeric(x))  
head(xc)

## Age BusinessTravel DailyRate Department DistanceFromHome Education  
## 1 41 3 1102 3 1 2  
## 2 49 2 279 2 8 1  
## 3 37 3 1373 2 2 2  
## 4 33 2 1392 2 3 4  
## 5 27 3 591 2 2 1  
## 6 32 2 1005 2 2 2  
## EducationField EnvironmentSatisfaction Gender HourlyRate JobInvolvement  
## 1 2 2 1 94 3  
## 2 2 3 2 61 2  
## 3 5 4 2 92 2  
## 4 2 4 1 56 3  
## 5 4 1 2 40 3  
## 6 2 4 2 79 3  
## JobLevel JobRole JobSatisfaction MaritalStatus MonthlyIncome MonthlyRate  
## 1 2 8 4 3 5993 19479  
## 2 2 7 2 2 5130 24907  
## 3 1 3 3 3 2090 2396  
## 4 1 7 3 2 2909 23159  
## 5 1 3 2 2 3468 16632  
## 6 1 3 4 3 3068 11864  
## NumCompaniesWorked OverTime PercentSalaryHike PerformanceRating  
## 1 8 2 11 1  
## 2 1 1 23 2  
## 3 6 2 15 1  
## 4 1 2 11 1  
## 5 9 1 12 1  
## 6 0 1 13 1  
## RelationshipSatisfaction StockOptionLevel TotalWorkingYears  
## 1 1 1 8  
## 2 4 2 10  
## 3 2 1 7  
## 4 3 1 8  
## 5 4 2 6  
## 6 3 1 8  
## TrainingTimesLastYear WorkLifeBalance YearsAtCompany YearsInCurrentRole  
## 1 0 1 6 4  
## 2 3 3 10 7  
## 3 3 3 0 0  
## 4 3 3 8 7  
## 5 3 3 2 2  
## 6 2 2 7 7  
## YearsSinceLastPromotion YearsWithCurrManager  
## 1 0 5  
## 2 1 7  
## 3 0 0  
## 4 3 0  
## 5 2 2  
## 6 3 6

# make all numeric  
xc\_att[] <- lapply(xc\_att, function(x) as.numeric(x))  
# reorder columns so attrition is last  
xc\_att <- xc\_att[,c(1, 3:31, 2)]  
head(xc\_att)

[results removed for ease of reading]

# Some parts of kmeans don't work well with NAs, so make sure those are gone  
colSums(is.na(xc))

[results removed for ease of reading]

## Depending on the data, we may need a scaled or transformed matrix. Make all three so we can visualize them.   
xc.m <- as.matrix(xc) # m stands for matrix  
xc.sm <-scale(xc.m) # sm for scaled matrix  
xc.tm <-t(xc.m) # tm for transformed matrix

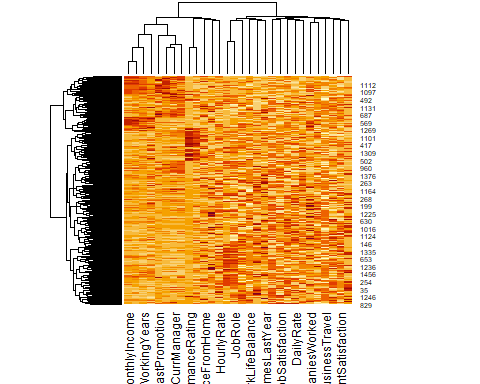
## visualize matrix  
### result: this matrix isn't useful. It needs to be scaled so that income isn't much higher.   
heatmap(xc.m)

[results removed for ease of reading]

## Visualize transformed matrix  
### result: there is a lot of variety in the data, but too many groups to be useful  
heatmap(xc.tm)

[results removed for ease of reading]

#colSums(is.na(xc.sm))  
heatmap(xc.sm)

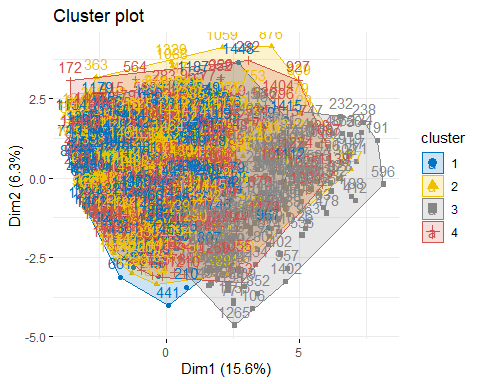


model\_xc4m <- kmeans(xc.m, 4)  
model\_xc4sm <- kmeans(xc.sm, 4)  
model\_xc4tm <- kmeans(xc.tm, 4)

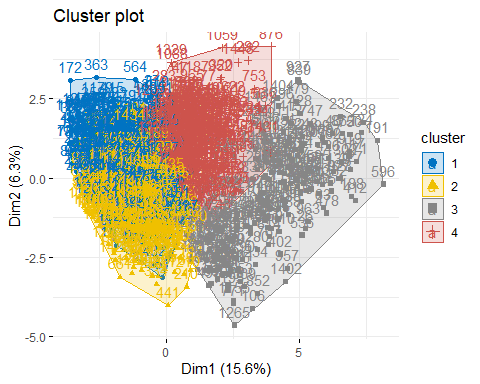
if("factoextra" %in% rownames(installed.packages()) == FALSE) {install.packages('factoextra') }  
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

## visualizing kmeans 4 groups with a cluster plot  
### because the numbers aren't scaled, the groups overlap  
fviz\_cluster(model\_xc4m, data = xc.m,  
 ellipse.type = "convex",  
 palette = "jco",  
 ggtheme = theme\_minimal())



### scaled matrix has three groups, but two overlap a lot  
fviz\_cluster(model\_xc4sm, data = xc.sm,  
 ellipse.type = "convex",  
 palette = "jco",  
 ggtheme = theme\_minimal())



### this isn't useful. Not using transformed matrix going forward.   
fviz\_cluster(model\_xc4tm, data = xc.tm,  
 ellipse.type = "convex",  
 palette = "jco",  
 ggtheme = theme\_minimal())

[results removed for ease of reading]

model\_xc6sm <- kmeans(xc.sm, 6)  
fviz\_cluster(model\_xc6sm, data = xc.sm,  
 ellipse.type = "convex",  
 palette = "jco",  
 ggtheme = theme\_minimal())

[results removed for ease of reading]

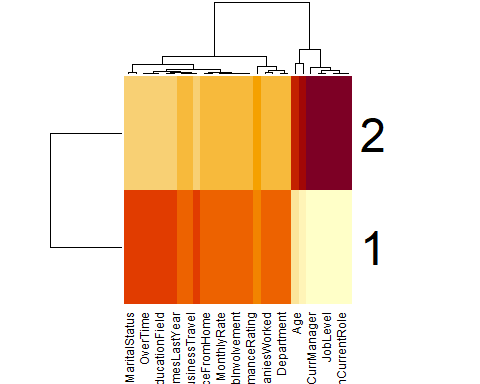
model\_xc3sm <- kmeans(xc.sm, 3)  
fviz\_cluster(model\_xc3sm, data = xc.sm,  
 ellipse.type = "convex",  
 palette = "jco",  
 ggtheme = theme\_minimal())

[results removed for ease of reading]

model\_xc2sm <- kmeans(xc.sm, 2)  
fviz\_cluster(model\_xc2sm, data = xc.sm,  
 ellipse.type = "convex",  
 palette = "jco",  
 ggtheme = theme\_minimal())

[results removed for ease of reading]

heatmap(model\_xc2sm$centers)



### this isn't useful because it is too coarse  
centers2 <- t(model\_xc2sm$centers)  
heatmap(centers2)

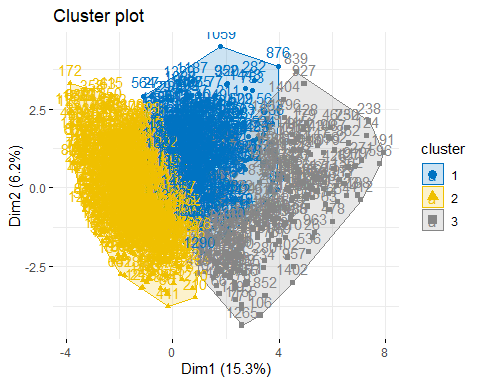
[results removed for ease of reading]

## does including attrition change the clusters?  
  
xc\_att.sm <-scale(as.matrix(xc\_att))  
model\_attsm2 <- kmeans(xc\_att.sm, 2)  
model\_attsm3 <- kmeans(xc\_att.sm, 3)  
model\_attsm4 <- kmeans(xc\_att.sm, 4)  
model\_attsm6 <- kmeans(xc\_att.sm, 6)

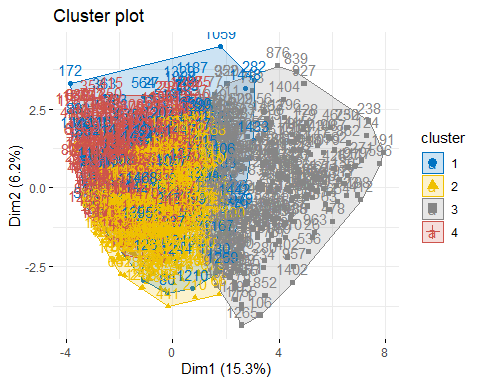
# no change at 2 clusters  
fviz\_cluster(model\_attsm2, data = xc\_att.sm,  
 ellipse.type = "convex",  
 palette = "jco",  
 ggtheme = theme\_minimal())

[results removed for ease of reading]

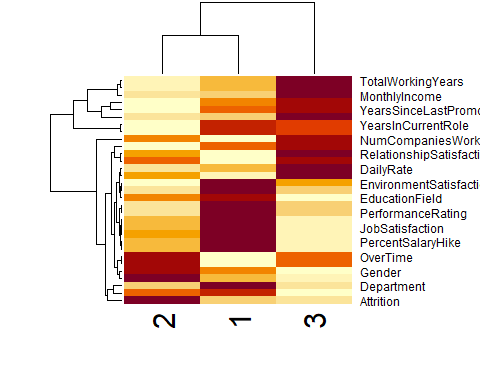
# with 3 clusters, there is some separation  
fviz\_cluster(model\_attsm3, data = xc\_att.sm,  
 ellipse.type = "convex",  
 palette = "jco",  
 ggtheme = theme\_minimal())



### with 4 clusters, there is too much overlap with three clusters  
### but one cluster is still separate  
fviz\_cluster(model\_attsm4, data = xc\_att.sm,  
 ellipse.type = "convex",  
 palette = "jco",  
 ggtheme = theme\_minimal())



centers\_att3 <- t(model\_attsm3$centers)  
heatmap(centers\_att3)



The centers for the 4-cluster model gives a clear heatmap.

### it looks like the cluster that was separate is for people with high job level, education, and business travel.  
### the three overlapping clusters differ in department, worklife balance, and overtime, among others  
### attrition seems high in one department and with worklife balance and training last year  
centers\_att4 <- t(model\_attsm4$centers)  
heatmap(centers\_att4)

Due to size restraints, parameters are missing in the Y-Axis.

head(xc\_att)

[results removed for ease of reading]

att\_YES <- xc\_att[which(xc\_att$Attrition == 2) , ]  
head(att\_YES)

[results removed for ease of reading]

str(att\_YES)

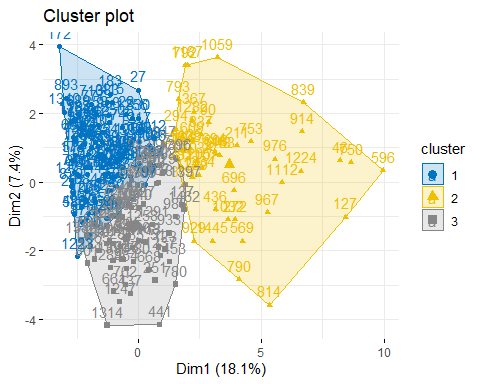
[results removed for ease of reading]

att\_YES.sm <-scale(as.matrix(att\_YES[,1:30]))  
head(att\_YES.sm)

[results removed for ease of reading]

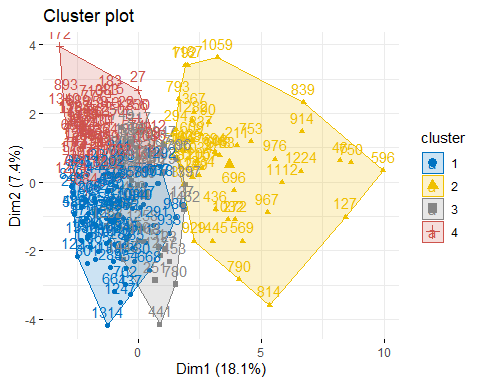
model\_YES3 <- kmeans(att\_YES.sm, 3)  
model\_YES4 <- kmeans(att\_YES.sm, 4)  
model\_YES5 <- kmeans(att\_YES.sm, 5)  
model\_YES6 <- kmeans(att\_YES.sm, 6)

fviz\_cluster(model\_YES3, data = att\_YES.sm,  
 ellipse.type = "convex",  
 palette = "jco",  
 ggtheme = theme\_minimal())

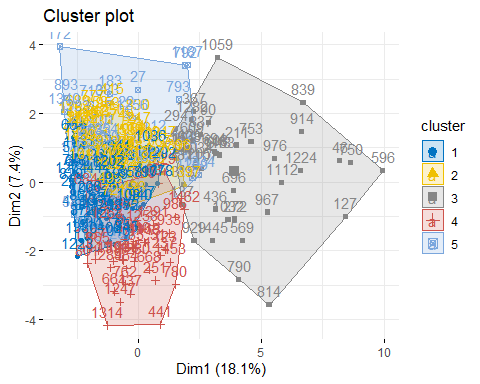


4 clusters appears to be the most useful Notice how few people left in the right group (cluster 1)

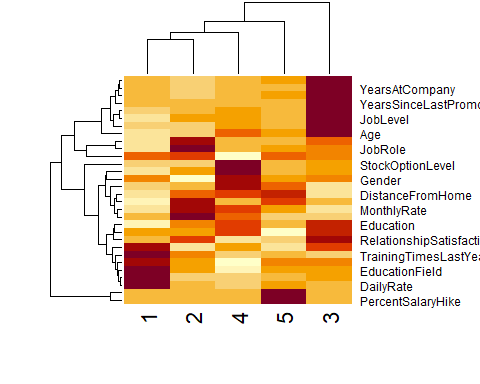
fviz\_cluster(model\_YES4, data = att\_YES.sm,  
 ellipse.type = "convex",  
 palette = "jco",  
 ggtheme = theme\_minimal())



fviz\_cluster(model\_YES5, data = att\_YES.sm,  
 ellipse.type = "convex",  
 palette = "jco",  
 ggtheme = theme\_minimal())



centers\_yes5 <- t(model\_YES5$centers)  
heatmap(centers\_yes5)



Looking at which attributes most distinguish between attrition = YES and attrition = NO.

model\_attsm4$centers

## Age BusinessTravel DailyRate Department DistanceFromHome  
## 1 -0.16138525 -0.01731657 -0.01343231 -0.063038984 0.06471734  
## 2 -0.08765187 -0.06904229 0.07443559 -0.074632334 0.04385912  
## 3 0.86386446 0.08893716 0.02001787 0.003015778 -0.06406643  
## 4 -0.42754145 0.02674738 -0.09533281 0.113676427 -0.03572065  
## Education EducationField EnvironmentSatisfaction Gender HourlyRate  
## 1 -0.09743924 -0.0114887865 -0.08737319 0.01236695 0.011564820  
## 2 0.05873249 0.0537880304 0.05573561 0.08633084 0.006110755  
## 3 0.17981694 -0.0838961318 0.10058393 -0.11490459 -0.045586880  
## 4 -0.15081571 0.0005167238 -0.09653060 -0.02671597 0.019455799  
## JobInvolvement JobLevel JobRole JobSatisfaction MaritalStatus  
## 1 -0.03922610 -0.2767719 -0.003659263 0.017140486 0.02586244  
## 2 0.08225515 -0.2905649 -0.059697647 0.026405206 -0.70216265  
## 3 -0.06617011 1.3426350 -0.179672737 -0.041459576 -0.05345210  
## 4 -0.03286534 -0.4717373 0.197034281 -0.009737053 0.84993344  
## MonthlyIncome MonthlyRate NumCompaniesWorked OverTime PercentSalaryHike  
## 1 -0.2870826 -0.04267619 -0.07329688 0.03325874 1.7960154  
## 2 -0.3042322 -0.07691641 0.04583451 -0.04334527 -0.3152389  
## 3 1.3582942 0.03619996 0.15520715 -0.01037723 -0.1872305  
## 4 -0.4620086 0.08410977 -0.12930062 0.04325254 -0.3021052  
## PerformanceRating RelationshipSatisfaction StockOptionLevel TotalWorkingYears  
## 1 2.3453529 -0.10755119 0.01074170 -0.2462420  
## 2 -0.4260850 -0.03444746 0.72037503 -0.2892699  
## 3 -0.1749515 0.07158572 -0.06193953 1.4144077  
## 4 -0.4260850 0.03857297 -0.80718411 -0.5369655  
## TrainingTimesLastYear WorkLifeBalance YearsAtCompany YearsInCurrentRole  
## 1 -0.008873963 -0.005164695 -0.2156679 -0.1023052  
## 2 0.016935248 -0.029498354 -0.2697115 -0.1927839  
## 3 -0.045156853 0.031065509 1.2537543 1.0094787  
## 4 0.015585064 0.015265263 -0.4615302 -0.4321103  
## YearsSinceLastPromotion YearsWithCurrManager Attrition  
## 1 -0.1570048 -0.1180193 0.06971126  
## 2 -0.2444599 -0.1994293 -0.26053141  
## 3 0.9821766 0.9740593 -0.22713505  
## 4 -0.3279604 -0.3925808 0.43308856

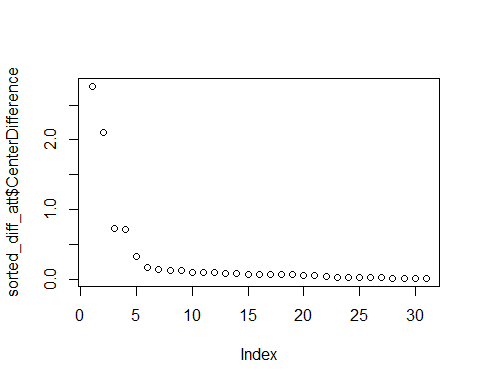
diff\_att <- data.frame(t(model\_attsm4$centers[1:2, ]))  
#rownames(diff\_att) <- c("Group1", "Group2")  
#difference.list <- abs(diff(diff\_att))  
diff\_att$CenterDifference <- round(abs(diff\_att$X1 - diff\_att$X2),2)  
diff\_att

## X1 X2 CenterDifference  
## Age -0.161385254 -0.087651873 0.07  
## BusinessTravel -0.017316570 -0.069042288 0.05  
## DailyRate -0.013432306 0.074435594 0.09  
## Department -0.063038984 -0.074632334 0.01  
## DistanceFromHome 0.064717341 0.043859117 0.02  
## Education -0.097439243 0.058732486 0.16  
## EducationField -0.011488786 0.053788030 0.07  
## EnvironmentSatisfaction -0.087373191 0.055735607 0.14  
## Gender 0.012366952 0.086330836 0.07  
## HourlyRate 0.011564820 0.006110755 0.01  
## JobInvolvement -0.039226103 0.082255146 0.12  
## JobLevel -0.276771865 -0.290564904 0.01  
## JobRole -0.003659263 -0.059697647 0.06  
## JobSatisfaction 0.017140486 0.026405206 0.01  
## MaritalStatus 0.025862440 -0.702162651 0.73  
## MonthlyIncome -0.287082649 -0.304232203 0.02  
## MonthlyRate -0.042676187 -0.076916411 0.03  
## NumCompaniesWorked -0.073296881 0.045834514 0.12  
## OverTime 0.033258735 -0.043345271 0.08  
## PercentSalaryHike 1.796015392 -0.315238856 2.11  
## PerformanceRating 2.345352910 -0.426085014 2.77  
## RelationshipSatisfaction -0.107551190 -0.034447458 0.07  
## StockOptionLevel 0.010741700 0.720375032 0.71  
## TotalWorkingYears -0.246242030 -0.289269867 0.04  
## TrainingTimesLastYear -0.008873963 0.016935248 0.03  
## WorkLifeBalance -0.005164695 -0.029498354 0.02  
## YearsAtCompany -0.215667862 -0.269711459 0.05  
## YearsInCurrentRole -0.102305223 -0.192783895 0.09  
## YearsSinceLastPromotion -0.157004769 -0.244459937 0.09  
## YearsWithCurrManager -0.118019334 -0.199429272 0.08  
## Attrition 0.069711264 -0.260531410 0.33

sorted\_diff\_att <- diff\_att[order(-diff\_att$CenterDifference),]  
sorted\_diff\_att[1:11, ]

## X1 X2 CenterDifference  
## PerformanceRating 2.34535291 -0.42608501 2.77  
## PercentSalaryHike 1.79601539 -0.31523886 2.11  
## MaritalStatus 0.02586244 -0.70216265 0.73  
## StockOptionLevel 0.01074170 0.72037503 0.71  
## Attrition 0.06971126 -0.26053141 0.33  
## Education -0.09743924 0.05873249 0.16  
## EnvironmentSatisfaction -0.08737319 0.05573561 0.14  
## JobInvolvement -0.03922610 0.08225515 0.12  
## NumCompaniesWorked -0.07329688 0.04583451 0.12  
## DailyRate -0.01343231 0.07443559 0.09  
## YearsInCurrentRole -0.10230522 -0.19278390 0.09

plot(sorted\_diff\_att$CenterDifference)



Looking at the group with the higest attrition and the group with the lowest attrition, the attributes with the biggest difference between those groups are:

* TotalWorkingYears
* NumCompaniesWorked
* JobLevel
* MonthlyIncome
* Education
* JobRole
* MaritalStatus
* StockOptionLevel
* JobInvolvement

#### Predictive Models

Based on the information gained from analyzing the data, we wanted to test the data against 5 different algorithms. The way we’re setting up our tests will be to use all parameters and then try a couple other subsets of parameters.

We will be running a few pilot tests on each model and on some of the subsets just to qualify where we should spend our time.

Once we have models and datasets that are bearing fruit, we will then run our full test. We will be using an alternate version of k-folds in which we will:

1. generate 5 samples based on set seeds
2. separate the data into 2/3 training and 1/3 testing
3. train the models based on the 2/3 training subset with the target variable as attrition
4. test the model against the 1/3 testing subset, removing the attrition label as needed
5. generate a confusion table for each seed based on the testing’s actual attrition
6. average the accuracy, precision yes, precision no, recall yes, and recall no for each

# Getting Set Up  
HR\_tree <- HR\_clean  
HR\_tree <- HR\_tree[,2:length(HR\_tree)]  
  
# Dataset 1/3  
# set Seed for randomizer to always pick the same  
seedNum1 <- 23  
seedNum2 <- 465  
seedNum3 <- 1  
seedNum4 <- 987  
seedNum5 <- 307  
  
set.seed(23)  
# Generate random sample of rows  
randIndex1 <- sample(1:nrow(HR\_clean))  
# Set 2/3 Cutpoint of total rows  
cutPoint <- floor(nrow(HR\_clean)\*2/3)  
# Create train data based on the 2/3 value  
trainData1 <- HR\_tree[randIndex1[1:cutPoint],]  
# Create test data based on the remaining 1/3  
testData1 <- HR\_tree[randIndex1[(cutPoint+1):length(randIndex1)],]  
  
# Dataset 2/3  
set.seed(465)  
# Generate random sample of rows  
randIndex2 <- sample(1:nrow(HR\_clean))  
  
# Dataset 2/3  
set.seed(1)  
# Generate random sample of rows  
randIndex3 <- sample(1:nrow(HR\_clean))

##### Decision Trees

Decision Trees is the first model we used. It is a modeling algorithm based applying decision points (nodes) that branch out toward either a computer outcome or toward more nodes for further branching. Decision Trees is a good starting point as it can tell us if a smaller subset of parameters is worth looking at as well as model on the whole parameter list.

To start, we’re running a decision tree with cp=0 and the default for minsplit and depth. This will give us a baseline to compare with.

# Function  
# Decision Tree Function:   
# First variable is putting in the seedNumber. I've set 5 variables labeled as seedNum1 - seedNum5  
# Second variable is whichever dataset that is being generated  
printDecision <- function(seedNum, dataSet, depth=5){  
 # set seed  
 set.seed(seedNum)  
 # Generate random sample of rows  
 randIndex <- sample(1:nrow(dataSet))  
 cutPoint <- floor(nrow(dataSet)\*2/3)  
 train <- dataSet[randIndex[1:cutPoint],]  
 test <- dataSet[randIndex[(cutPoint+1):length(randIndex)],]  
 decisionTree <- rpart(Attrition ~ ., data = train, method="class", control=rpart.control(cp=0, minsplit = 5, maxdepth = depth))  
 summary(decisionTree)  
 # plot number of splits  
 rpart.plot(decisionTree, tweak=1.6)  
 # Predictions  
 predicted <- predict(decisionTree, test, type="class")  
 print(summary(predicted))  
 print(table(predictedAttrition=predicted, actualAttrition=test$Attrition))  
 set.seed(NULL)  
}

library(rpart)  
library(rattle)  
library(rpart.plot)  
basicTree <- rpart(Attrition ~ ., data = trainData1, method="class", control=rpart.control(cp=0))  
summary(basicTree)

## Call:  
## rpart(formula = Attrition ~ ., data = trainData1, method = "class",   
## control = rpart.control(cp = 0))  
## n= 980   
##   
## CP nsplit rel error xerror xstd  
## 1 0.059375000 0 1.00000 1.00000 0.07231592  
## 2 0.031250000 2 0.88125 0.87500 0.06846532  
## 3 0.025000000 4 0.81875 0.90625 0.06946951  
## 4 0.020833333 5 0.79375 0.93750 0.07044524  
## 5 0.018750000 8 0.73125 0.94375 0.07063708  
## 6 0.012500000 9 0.71250 0.93750 0.07044524  
## 7 0.008333333 14 0.65000 0.99375 0.07213352  
## 8 0.006250000 17 0.62500 1.01250 0.07267769  
## 9 0.002083333 20 0.60625 1.08750 0.07476686  
## 10 0.000000000 23 0.60000 1.13750 0.07608589  
##   
## Variable importance  
## MonthlyIncome OverTime TotalWorkingYears   
## 13 10 6   
## JobRole DailyRate DistanceFromHome   
## 6 6 5   
## YearsAtCompany MonthlyRate MaritalStatus   
## 4 4 4   
## YearsWithCurrManager Department EducationField   
## 4 4 4   
## YearsInCurrentRole EnvironmentSatisfaction HourlyRate   
## 4 4 4   
## Age StockOptionLevel JobLevel   
## 3 3 3   
## BusinessTravel JobInvolvement YearsSinceLastPromotion   
## 2 2 1   
## JobSatisfaction Gender NumCompaniesWorked   
## 1 1 1   
## PercentSalaryHike   
## 1   
##

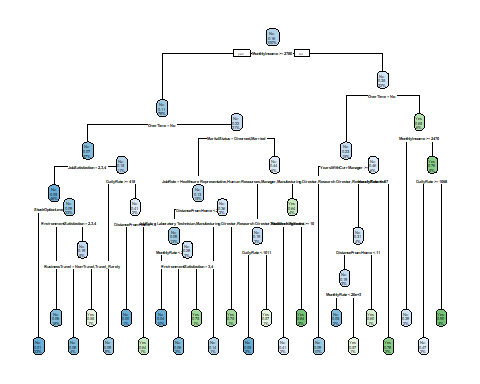
#predict the test dataset using the model for train tree No. 1  
basicPredict <- predict(basicTree, testData1, type="class")  
#plot number of splits  
summary(basicPredict)

## No Yes   
## 432 58

table(predictedAttrition=basicPredict, actualAttrition=testData1$Attrition)

## actualAttrition  
## predictedAttrition No Yes  
## No 375 57  
## Yes 38 20

#Disputed Prediction  
rpart.plot(basicTree, tweak=1.6)



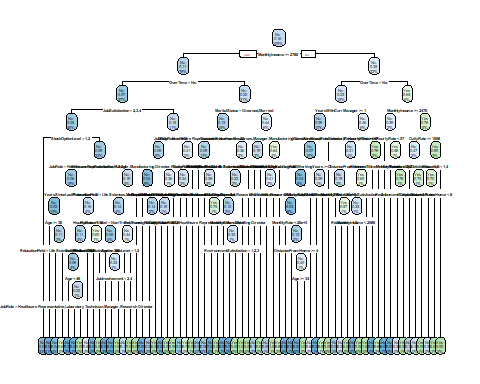
In looking at the table, we can calculate accuracy: 395/490 = ~.806

We’ll run the model on increase to 10 depth and minsplit 5. Minsplit is that there must be a minimum of 5 results in each separation and depth is how many divisions there are.

advancedTree <- printDecision(seedNum1, HR\_tree, 10)

## Call:  
## rpart(formula = Attrition ~ ., data = train, method = "class",   
## control = rpart.control(cp = 0, minsplit = 5, maxdepth = depth))  
## n= 980   
##   
## CP nsplit rel error xerror xstd  
## 1 0.059375000 0 1.00000 1.00000 0.07231592  
## 2 0.031250000 2 0.88125 0.90000 0.06927099  
## 3 0.025000000 4 0.81875 0.93750 0.07044524  
## 4 0.020833333 5 0.79375 0.91875 0.06986315  
## 5 0.015625000 10 0.67500 0.94375 0.07063708  
## 6 0.012500000 14 0.61250 0.93750 0.07044524  
## 7 0.010416667 26 0.46250 0.99375 0.07213352  
## 8 0.009375000 30 0.41875 1.03125 0.07321292  
## 9 0.006250000 37 0.35000 1.07500 0.07442809  
## 10 0.004166667 55 0.23125 1.17500 0.07703862  
## 11 0.003125000 58 0.21875 1.18125 0.07719446  
## 12 0.000000000 64 0.20000 1.20625 0.07780957  
##

[results removed for ease of reading]



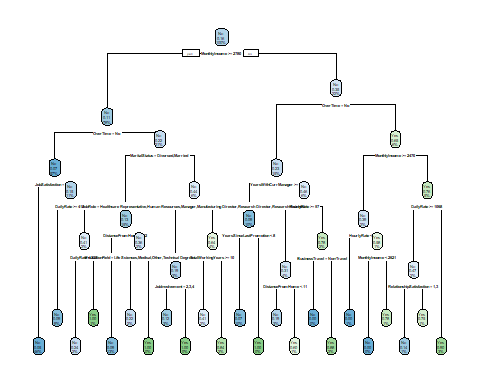
## No Yes   
## 428 62   
## actualAttrition  
## predictedAttrition No Yes  
## No 372 56  
## Yes 41 21

This tells us that the accuracy is 393/490 = ~.802

We’ll run the model on 5 depth now.

# Increase minSplit and maxDepth  
advancedTree <- printDecision(seedNum1, HR\_tree)

[results removed for ease of reading]



## No Yes   
## 445 45   
## actualAttrition  
## predictedAttrition No Yes  
## No 386 59  
## Yes 27 18

Accuracy: 406/490 = ~.829

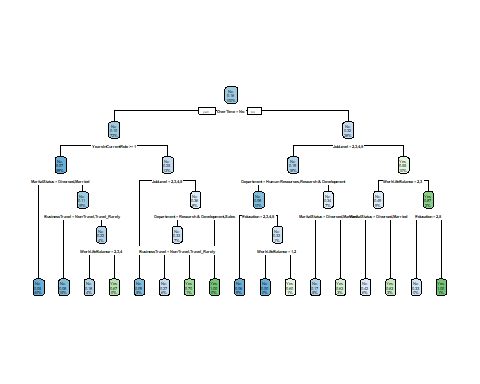
5 depth seems to be doing best.

Based on previous information from KMeans and from Apriori, let’s select/remove fields. Selecting:

* Attrition
* BusinessTravel
* Department
* Education
* JobLevel
* MaritalStatus
* MonthlyIncome
* OverTime
* WorkLifeBalance
* YearsWithCurrManager
* YearsInCurrentRole

treeSpecific <- data.frame(HR\_tree$Attrition, HR\_tree$BusinessTravel, HR\_tree$Department, HR\_tree$Education, HR\_tree$JobLevel, HR\_tree$MaritalStatus, HR\_tree$OverTime, HR\_tree$WorkLifeBalance, HR\_tree$YearsInCurrentRole, HR\_tree$YearsWithCurrManager )  
  
# Picking specific attributes based on what the previous analysis  
colnames(treeSpecific) <- c("Attrition","BusinessTravel","Department","Education","JobLevel","MaritalStatus","OverTime","WorkLifeBalance","YearsWithCurrManager","YearsInCurrentRole")  
  
specificTree <- printDecision(seedNum1, treeSpecific)

[results removed for ease of reading]



## No Yes   
## 445 45   
## actualAttrition  
## predictedAttrition No Yes  
## No 391 54  
## Yes 22 23

Predicted Accuracy: 414/490 = ~ .845

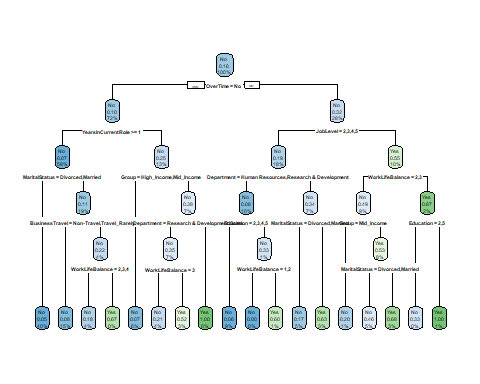
We’ll add in income as it was one of the parameters as well, but it needs to be folded into the same discrete form.

# Determining percentiles  
Percentile\_00 = min(HR\_tree$MonthlyIncome)  
Percentile\_33 = quantile(HR\_tree$MonthlyIncome, 0.33333)  
Percentile\_67 = quantile(HR\_tree$MonthlyIncome, 0.66667)  
Percentile\_100 = max(HR\_tree$MonthlyIncome)  
  
# Values  
HR.Bind = rbind(Percentile\_00, Percentile\_33, Percentile\_67, Percentile\_100)  
dimnames(HR.Bind)[[2]] = "Value"  
HR.Bind

## Value  
## Percentile\_00 1009.000  
## Percentile\_33 3631.647  
## Percentile\_67 6528.735  
## Percentile\_100 19999.000

# Grouping  
treeIncome <- treeSpecific  
treeIncome$income <- HR\_tree$MonthlyIncome  
treeIncome$Group[treeIncome$income >= Percentile\_00 & treeIncome$income < Percentile\_33] = "Low\_Income"  
treeIncome$Group[treeIncome$income >= Percentile\_33 & treeIncome$income < Percentile\_67] = "Mid\_Income"  
treeIncome$Group[treeIncome$income >= Percentile\_67 & treeIncome$income <= Percentile\_100] = "High\_Income"  
treeIncome$income <- NULL  
  
incomeTree <- printDecision(seedNum1, treeIncome)

[results removed for ease of reading]



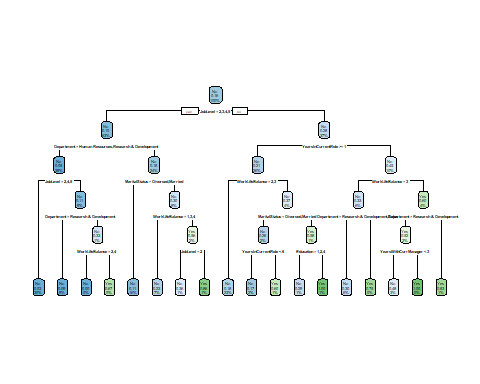
## No Yes   
## 443 47   
## actualAttrition  
## predictedAttrition No Yes  
## No 388 55  
## Yes 25 22

410/490 - worse off.

We’ll use the specific 10 fields and remove overtime and business travel because they are correlated with worklifebalance.

# Grouping  
reducedFields <- treeSpecific  
reducedFields$Group <- NULL  
reducedFields$OverTime <- NULL  
reducedFields$BusinessTravel<- NULL  
  
printDecision(seedNum1, reducedFields)

[results removed for ease of reading]



## No Yes   
## 480 10   
## actualAttrition  
## predictedAttrition No Yes  
## No 406 74  
## Yes 7 3

409/490 This winds up being worse.

We will then run our final tests and aggregate them into our results dataframe. This will be depth 5 and on both the full parameter set as well as the specific 10.

confusionTable <- function(seedNum, dataSet){  
 # set seed  
 set.seed(seedNum)  
 # Generate random sample of rows  
 randIndex <- sample(1:nrow(dataSet))  
 cutPoint <- floor(nrow(dataSet)\*2/3)  
 train <- dataSet[randIndex[1:cutPoint],]  
 test <- dataSet[randIndex[(cutPoint+1):length(randIndex)],]  
 decisionTree <- rpart(Attrition ~ ., data = train, method="class", control=rpart.control(cp=0, minsplit = 5, maxdepth = 5))  
 predicted <- predict(decisionTree, test, type="class")  
 set.seed(NULL)  
 return(table(predictedAttrition=predicted, actualAttrition=test$Attrition))  
}  
  
tableCalc <- function(table){  
 calcTable <- as.data.frame(table)  
 accuracy <- (calcTable[which(calcTable$predictedAttrition=="Yes" & calcTable$actualAttrition=="Yes"), 3] + calcTable[which(calcTable$predictedAttrition=="No" && calcTable$actualAttrition=="No"), 3])/sum(calcTable$Freq)  
 precisionYes <- (calcTable[which(calcTable$predictedAttrition=="Yes" & calcTable$actualAttrition=="Yes"), 3])/(calcTable[which(calcTable$predictedAttrition=="Yes" & calcTable$actualAttrition=="Yes"),3] + calcTable[which(calcTable$predictedAttrition=="Yes" & calcTable$actualAttrition=="No"),3])  
 precisionNo <- (calcTable[which(calcTable$predictedAttrition=="No" & calcTable$actualAttrition=="No"), 3])/(calcTable[which(calcTable$predictedAttrition=="No" & calcTable$actualAttrition=="Yes"),3] + calcTable[which(calcTable$predictedAttrition=="No" & calcTable$actualAttrition=="No"),3])  
 recallYes <- (calcTable[which(calcTable$predictedAttrition=="Yes" & calcTable$actualAttrition=="Yes"), 3])/(calcTable[which(calcTable$predictedAttrition=="Yes" & calcTable$actualAttrition=="Yes"),3] + calcTable[which(calcTable$predictedAttrition=="No" & calcTable$actualAttrition=="Yes"),3])  
 recallNo <- (calcTable[which(calcTable$predictedAttrition=="No" & calcTable$actualAttrition=="No"), 3])/(calcTable[which(calcTable$predictedAttrition=="Yes" & calcTable$actualAttrition=="No"),3] + calcTable[which(calcTable$predictedAttrition=="No" & calcTable$actualAttrition=="No"),3])  
 dataFrame <- data.frame(accuracy,precisionYes,precisionNo,recallYes,recallNo)  
 return(dataFrame)  
}  
  
averageTableCalc <- function(dataFrame){  
 avgAccuracy <- mean(dataFrame$accuracy)  
 avgPrecisionYes <- mean(dataFrame$precisionYes)  
 avgPrecisionNo <- mean(dataFrame$precisionNo)  
 avgRecallYes <- mean(dataFrame$recallYes)  
 avgRecallNo <- mean(dataFrame$recallNo)  
 newDF <- data.frame(avgAccuracy, avgPrecisionYes, avgPrecisionNo, avgRecallYes, avgRecallNo)  
 return(newDF)  
}  
  
completeTreeFunc <- function(dataSet){  
 treeTable1 <- confusionTable(seedNum1, dataSet)  
 treeTable2 <- confusionTable(seedNum2, dataSet)  
 treeTable3 <- confusionTable(seedNum3, dataSet)  
 treeTable4 <- confusionTable(seedNum4, dataSet)  
 treeTable5 <- confusionTable(seedNum5, dataSet)  
   
 treeTableCalc1 <- tableCalc(treeTable1)  
 treeTableCalc2 <- tableCalc(treeTable2)  
 treeTableCalc3 <- tableCalc(treeTable3)  
 treeTableCalc4 <- tableCalc(treeTable4)  
 treeTableCalc5 <- tableCalc(treeTable5)  
   
 treeTableCalc <- data.frame(rbind(as.matrix(treeTableCalc1),as.matrix(treeTableCalc2),as.matrix(treeTableCalc3),as.matrix(treeTableCalc4),as.matrix(treeTableCalc5)))  
   
 avgTreeCalc <- averageTableCalc(treeTableCalc)  
 print(avgTreeCalc)  
}

hrTree <- completeTreeFunc(HR\_tree)

[results removed for ease of reading]

hrTree$type <- "decisionTrees\_hrTree"  
hrTreeSpecific <- completeTreeFunc(treeSpecific)

[results removed for ease of reading]

hrTreeSpecific$type <- "decisionTrees\_treeSpecific"  
hrTreeIncome <- completeTreeFunc(treeIncome)

[results removed for ease of reading]

hrTreeIncome$type <- "decisionTrees\_treeIncome"  
hrTreeReduced <- completeTreeFunc(reducedFields)

[results removed for ease of reading]

hrTreeReduced$type <- "decisionTrees\_treeReduced"  
completeModels <- rbind(hrTree, hrTreeSpecific, hrTreeIncome, hrTreeReduced)  
completeModels

[results removed for ease of reading]

##### Support Vector Machines

The next one we ran is Support Vector Machines (SVM), a supervised learning model. It looks at the data and decides on how to create a linear plane that would cut across the dimensions and separate betwen classes. There are two parameters we will be adjusting:

1. the kernel, which is how it identifies the type of dimensional perspective
2. the cost, which is how it punishes for an incorrect assignment

library(kernlab)  
library(e1071)  
  
printSVM <- function(seedNum, dataSet, kernelType="radial", cost=1){  
 # set seed  
 set.seed(seedNum)  
 # Generate random sample of rows  
 randIndex <- sample(1:nrow(dataSet))  
 cutPoint <- floor(nrow(dataSet)\*2/3)  
 train <- dataSet[randIndex[1:cutPoint],]  
 test <- dataSet[randIndex[(cutPoint+1):length(randIndex)],]  
 svmModel <- svm(Attrition ~ ., data = train, kernel=kernelType, cost=cost)  
 # Predictions  
 predicted <- predict(svmModel, test, type="votes")  
 print(table(predictedAttrition=predicted, actualAttrition=test$Attrition))  
 set.seed(NULL)  
}

kernelName <- "radial"  
print(kernelName)

## [1] "radial"

dataFrame <- HR\_tree  
printSVM(seedNum1, dataFrame, kernelName, cost=1)

## actualAttrition  
## predictedAttrition No Yes  
## No 413 76  
## Yes 0 1

printSVM(seedNum1, dataFrame, kernelName, cost=.7)  
printSVM(seedNum1, dataFrame, kernelName, cost=.5)  
printSVM(seedNum1, dataFrame, kernelName, cost=.3)  
printSVM(seedNum1, dataFrame, kernelName, cost=.1)  
printSVM(seedNum2, dataFrame, kernelName, cost=1)  
printSVM(seedNum2, dataFrame, kernelName, cost=.7)  
printSVM(seedNum2, dataFrame, kernelName, cost=.5)  
printSVM(seedNum2, dataFrame, kernelName, cost=.3)  
printSVM(seedNum2, dataFrame, kernelName, cost=.1)  
printSVM(seedNum3, dataFrame, kernelName, cost=1)  
printSVM(seedNum3, dataFrame, kernelName, cost=.7)  
printSVM(seedNum3, dataFrame, kernelName, cost=.5)  
printSVM(seedNum3, dataFrame, kernelName, cost=.3)  
printSVM(seedNum3, dataFrame, kernelName, cost=.1)

Radial seems to be only guessing no to attrition.

kernelName <- "sigmoid"  
print(kernelName)

## [1] "sigmoid"

dataFrame <- HR\_tree  
printSVM(seedNum1, dataFrame, kernelName, cost=1)

## actualAttrition  
## predictedAttrition No Yes  
## No 413 77  
## Yes 0 0

printSVM(seedNum1, dataFrame, kernelName, cost=.7)  
printSVM(seedNum1, dataFrame, kernelName, cost=.5)  
printSVM(seedNum1, dataFrame, kernelName, cost=.3)  
printSVM(seedNum1, dataFrame, kernelName, cost=.1)  
printSVM(seedNum2, dataFrame, kernelName, cost=1)  
printSVM(seedNum2, dataFrame, kernelName, cost=.7)  
printSVM(seedNum2, dataFrame, kernelName, cost=.5)  
printSVM(seedNum2, dataFrame, kernelName, cost=.3)  
printSVM(seedNum2, dataFrame, kernelName, cost=.1)  
printSVM(seedNum3, dataFrame, kernelName, cost=1)  
printSVM(seedNum3, dataFrame, kernelName, cost=.7)  
printSVM(seedNum3, dataFrame, kernelName, cost=.5)  
printSVM(seedNum3, dataFrame, kernelName, cost=.3)  
printSVM(seedNum3, dataFrame, kernelName, cost=.1)

Sigmoid seems to be only guessing no to attrition.

kernelName <- "polynomial"  
print(kernelName)

## [1] "polynomial"

dataFrame <- HR\_tree  
printSVM(seedNum1, dataFrame, kernelName, cost=1)

## actualAttrition  
## predictedAttrition No Yes  
## No 413 77  
## Yes 0 0

printSVM(seedNum1, dataFrame, kernelName, cost=.7)  
printSVM(seedNum1, dataFrame, kernelName, cost=.5)  
printSVM(seedNum1, dataFrame, kernelName, cost=.3)  
printSVM(seedNum1, dataFrame, kernelName, cost=.1)  
printSVM(seedNum2, dataFrame, kernelName, cost=1)  
printSVM(seedNum2, dataFrame, kernelName, cost=.7)  
printSVM(seedNum2, dataFrame, kernelName, cost=.5)  
printSVM(seedNum2, dataFrame, kernelName, cost=.3)  
printSVM(seedNum2, dataFrame, kernelName, cost=.1)  
printSVM(seedNum3, dataFrame, kernelName, cost=1)  
printSVM(seedNum3, dataFrame, kernelName, cost=.7)  
printSVM(seedNum3, dataFrame, kernelName, cost=.5)  
printSVM(seedNum3, dataFrame, kernelName, cost=.3)  
printSVM(seedNum3, dataFrame, kernelName, cost=.1)

Polynomial seems to be only guessing no to attrition.

kernelName <- "linear"  
print(kernelName)

## [1] "linear"

dataFrame <- HR\_tree  
printSVM(seedNum1, dataFrame, kernelName, cost=1)

## actualAttrition  
## predictedAttrition No Yes  
## No 390 37  
## Yes 23 40

printSVM(seedNum1, dataFrame, kernelName, cost=.7)

## actualAttrition  
## predictedAttrition No Yes  
## No 391 37  
## Yes 22 40

printSVM(seedNum1, dataFrame, kernelName, cost=.5)

## actualAttrition  
## predictedAttrition No Yes  
## No 393 39  
## Yes 20 38

printSVM(seedNum1, dataFrame, kernelName, cost=.3)

## actualAttrition  
## predictedAttrition No Yes  
## No 397 42  
## Yes 16 35

printSVM(seedNum1, dataFrame, kernelName, cost=.1)

## actualAttrition  
## predictedAttrition No Yes  
## No 405 45  
## Yes 8 32

[results removed for ease of reading]

Linear is providing some information. We’ll leave cost at .5 as there seems to be quite a variance across the costs and the seedNum.

kernelName <- "linear"  
print(kernelName)

## [1] "linear"

print("treeSpecific")

## [1] "treeSpecific"

dataFrame <- treeSpecific  
printSVM(seedNum1, dataFrame, kernelName, cost=1)

## actualAttrition  
## predictedAttrition No Yes  
## No 413 77  
## Yes 0 0

printSVM(seedNum1, dataFrame, kernelName, cost=.7)  
printSVM(seedNum1, dataFrame, kernelName, cost=.5)  
printSVM(seedNum1, dataFrame, kernelName, cost=.3)  
printSVM(seedNum1, dataFrame, kernelName, cost=.1)  
printSVM(seedNum2, dataFrame, kernelName, cost=1)  
printSVM(seedNum2, dataFrame, kernelName, cost=.7)  
printSVM(seedNum2, dataFrame, kernelName, cost=.5)  
printSVM(seedNum2, dataFrame, kernelName, cost=.3)  
printSVM(seedNum2, dataFrame, kernelName, cost=.1)  
printSVM(seedNum3, dataFrame, kernelName, cost=1)  
printSVM(seedNum3, dataFrame, kernelName, cost=.7)  
printSVM(seedNum3, dataFrame, kernelName, cost=.5)  
printSVM(seedNum3, dataFrame, kernelName, cost=.3)  
printSVM(seedNum3, dataFrame, kernelName, cost=.1)

Using the specific 10 parameters, it also starts guessing no only.

We will then use linear across all parameters at .5 cost and add the average of the 5 seeds to our final dataframe.

confusionTableSVM <- function(seedNum, dataSet){  
 # set seed  
 set.seed(seedNum)  
 # Generate random sample of rows  
 randIndex <- sample(1:nrow(dataSet))  
 cutPoint <- floor(nrow(dataSet)\*2/3)  
 train <- dataSet[randIndex[1:cutPoint],]  
 test <- dataSet[randIndex[(cutPoint+1):length(randIndex)],]  
 algorithm <- svm(Attrition ~ ., data = train, kernel="linear", cost=.5)  
 predicted <- predict(algorithm, test, type="class")  
 set.seed(NULL)  
 return(table(predictedAttrition=predicted, actualAttrition=test$Attrition))  
}  
  
  
completeSVMFunc <- function(dataSet){  
 table1 <- confusionTableSVM(seedNum1, dataSet)  
 table2 <- confusionTableSVM(seedNum2, dataSet)  
 table3 <- confusionTableSVM(seedNum3, dataSet)  
 table4 <- confusionTableSVM(seedNum4, dataSet)  
 table5 <- confusionTableSVM(seedNum5, dataSet)  
   
 tableCalc1 <- tableCalc(table1)  
 tableCalc2 <- tableCalc(table2)  
 tableCalc3 <- tableCalc(table3)  
 tableCalc4 <- tableCalc(table4)  
 tableCalc5 <- tableCalc(table5)  
   
 tableCalc <- data.frame(rbind(as.matrix(tableCalc1),as.matrix(tableCalc2),as.matrix(tableCalc3),as.matrix(tableCalc4),as.matrix(tableCalc5)))  
   
 avgTableCalc <- averageTableCalc(tableCalc)  
 print(avgTableCalc)  
}

hrSVM <- completeSVMFunc(HR\_tree)

[results removed for ease of reading]

hrSVM$type <- "svm\_hrTree"  
completeModels <- rbind(completeModels, hrSVM)  
completeModels

[results removed for ease of reading]

##### Naive Bayes

Our next algorithm is Naive Bayes: a machine learning classifier that’s based on Bayes theorem of dependent probability. It’s Naive however because it assumes independence between the variables, meaning that they are not probabilistically linked. For our system, that would mean that one pixel is not intrinsically linked to another pixel.

We will be running it with changing the laplace number to see if it improves.

#  
printNB <- function(seedNum, dataSet, laplaceNum=1){  
 # set seed  
 set.seed(seedNum)  
 # Generate random sample of rows  
 randIndex <- sample(1:nrow(dataSet))  
 cutPoint <- floor(nrow(dataSet)\*2/3)  
 train <- dataSet[randIndex[1:cutPoint],]  
 test <- dataSet[randIndex[(cutPoint+1):length(randIndex)],]  
 model=naiveBayes(Attrition~., data = train, laplace = laplaceNum, na.action = na.pass)  
 # Predictions  
 predicted <- predict(model, test)  
 print(table(predictedAttrition=predicted, actualAttrition=test$Attrition))  
 set.seed(NULL)  
}

printNB(seedNum1, HR\_tree)

## actualAttrition  
## predictedAttrition No Yes  
## No 342 26  
## Yes 71 51

printNB(seedNum1, HR\_tree, 2)

## actualAttrition  
## predictedAttrition No Yes  
## No 343 25  
## Yes 70 52

printNB(seedNum1, HR\_tree, 5)

## actualAttrition  
## predictedAttrition No Yes  
## No 347 26  
## Yes 66 51

printNB(seedNum1, HR\_tree, 10)

## actualAttrition  
## predictedAttrition No Yes  
## No 352 27  
## Yes 61 50

printNB(seedNum1, HR\_tree, 15)

## actualAttrition  
## predictedAttrition No Yes  
## No 358 29  
## Yes 55 48

More seeds were tested, but not much difference in them.

printNB(seedNum2, HR\_tree)  
printNB(seedNum2, HR\_tree, 2)  
printNB(seedNum2, HR\_tree, 5)  
printNB(seedNum2, HR\_tree, 10)  
printNB(seedNum2, HR\_tree, 15)  
printNB(seedNum3, HR\_tree)  
printNB(seedNum3, HR\_tree, 2)  
printNB(seedNum3, HR\_tree, 5)  
printNB(seedNum3, HR\_tree, 10)  
printNB(seedNum3, HR\_tree, 15)

printNB(seedNum1, treeSpecific)

## actualAttrition  
## predictedAttrition No Yes  
## No 396 50  
## Yes 17 27

printNB(seedNum1, treeSpecific, 2)

[results removed for ease of reading]

printNB(seedNum1, treeSpecific, 5)

[results removed for ease of reading]

printNB(seedNum1, treeSpecific, 10)

[results removed for ease of reading]

printNB(seedNum1, treeSpecific, 15)

[results removed for ease of reading]

More seeds were tested, but not much difference in them.

printNB(seedNum2, treeSpecific)  
printNB(seedNum2, treeSpecific, 2)  
printNB(seedNum2, treeSpecific, 5)  
printNB(seedNum2, treeSpecific, 10)  
printNB(seedNum2, treeSpecific, 15)  
printNB(seedNum3, treeSpecific)  
printNB(seedNum3, treeSpecific, 2)  
printNB(seedNum3, treeSpecific, 5)  
printNB(seedNum3, treeSpecific, 10)  
printNB(seedNum3, treeSpecific, 15)

Both datasets seem like potentials so we will add both, but increasing Laplace does not make a tremendous jump until 15. But that also causes overfitting, so we will leave at laplace 1.

confusionTableNB <- function(seedNum, dataSet, laplaceNum=1){  
 # set seed  
 set.seed(seedNum)  
 # Generate random sample of rows  
 randIndex <- sample(1:nrow(dataSet))  
 cutPoint <- floor(nrow(dataSet)\*2/3)  
 train <- dataSet[randIndex[1:cutPoint],]  
 test <- dataSet[randIndex[(cutPoint+1):length(randIndex)],]  
 algorithm <- naiveBayes(Attrition~., data = train, laplace = laplaceNum, na.action = na.pass)  
 predicted <- predict(algorithm, test, type="class")  
 set.seed(NULL)  
 return(table(predictedAttrition=predicted, actualAttrition=test$Attrition))  
}  
  
  
completeNBFunc <- function(dataSet, laplaceNum=1){  
 table1 <- confusionTableNB(seedNum1, dataSet, laplaceNum)  
 table2 <- confusionTableNB(seedNum2, dataSet, laplaceNum)  
 table3 <- confusionTableNB(seedNum3, dataSet, laplaceNum)  
 table4 <- confusionTableNB(seedNum4, dataSet, laplaceNum)  
 table5 <- confusionTableNB(seedNum5, dataSet, laplaceNum)  
   
 tableCalc1 <- tableCalc(table1)  
 tableCalc2 <- tableCalc(table2)  
 tableCalc3 <- tableCalc(table3)  
 tableCalc4 <- tableCalc(table4)  
 tableCalc5 <- tableCalc(table5)  
   
 tableCalc <- data.frame(rbind(as.matrix(tableCalc1),as.matrix(tableCalc2),as.matrix(tableCalc3),as.matrix(tableCalc4),as.matrix(tableCalc5)))  
   
 avgTableCalc <- averageTableCalc(tableCalc)  
 print(avgTableCalc)  
}

nbModel <- completeNBFunc(HR\_tree, 1)

[results removed for ease of reading]

nbModel$type <- "nb\_hrTree"  
completeModels <- rbind(completeModels, nbModel)  
nb2Model <- completeNBFunc(treeSpecific, 1)

[results removed for ease of reading]

nb2Model$type <- "nb\_treeSpecific"  
completeModels <- rbind(completeModels, nb2Model)  
completeModels

[results removed for ease of reading]

##### Random Forest

Our next algorithm is Random Forest: an ensemble learning algorithm that produces multiple trees and outputs the class that is the most often selected class. We will be increasing trees from 3 to 25. As the more trees we add in, the more overfitting may occur.

library(randomForest)

printRF <- function(seedNum, dataSet, trees=3){  
 # set seed  
 set.seed(seedNum)  
 # Generate random sample of rows  
 randIndex <- sample(1:nrow(dataSet))  
 cutPoint <- floor(nrow(dataSet)\*2/3)  
 train <- dataSet[randIndex[1:cutPoint],]  
 test <- dataSet[randIndex[(cutPoint+1):length(randIndex)],]  
 model=randomForest(Attrition~., data = train, ntree=trees)  
 # Predictions  
 predicted <- predict(model, test, type=c("class"))  
 print(table(predictedAttrition=predicted, actualAttrition=test$Attrition))  
 set.seed(NULL)  
}

printRF(seedNum1, HR\_tree)

## actualAttrition  
## predictedAttrition No Yes  
## No 364 54  
## Yes 49 23

printRF(seedNum1, HR\_tree, 5)

## actualAttrition  
## predictedAttrition No Yes  
## No 389 60  
## Yes 24 17

printRF(seedNum1, HR\_tree, 10)

## actualAttrition  
## predictedAttrition No Yes  
## No 399 60  
## Yes 14 17

printRF(seedNum1, HR\_tree, 15)

## actualAttrition  
## predictedAttrition No Yes  
## No 408 62  
## Yes 5 15

printRF(seedNum1, HR\_tree, 25)

## actualAttrition  
## predictedAttrition No Yes  
## No 404 64  
## Yes 9 13

More seeds were tested, but not much difference in them.

printRF(seedNum2, HR\_tree)  
printRF(seedNum2, HR\_tree, 5)  
printRF(seedNum2, HR\_tree, 10)  
printRF(seedNum2, HR\_tree, 15)  
printRF(seedNum2, HR\_tree, 25)  
printRF(seedNum3, HR\_tree)  
printRF(seedNum3, HR\_tree, 5)  
printRF(seedNum3, HR\_tree, 10)  
printRF(seedNum3, HR\_tree, 15)  
printRF(seedNum3, HR\_tree, 25)

printRF(seedNum1, treeSpecific)

## actualAttrition  
## predictedAttrition No Yes  
## No 382 56  
## Yes 31 21

printRF(seedNum1, treeSpecific, 5)

[results removed for ease of reading]

printRF(seedNum1, treeSpecific, 10)

[results removed for ease of reading]

printRF(seedNum1, treeSpecific, 15)

[results removed for ease of reading]

printRF(seedNum1, treeSpecific, 25)

[results removed for ease of reading]

More seeds were tested, but not much difference in them.

printRF(seedNum2, treeSpecific)  
printRF(seedNum2, treeSpecific, 5)  
printRF(seedNum2, treeSpecific, 10)  
printRF(seedNum2, treeSpecific, 15)  
printRF(seedNum2, treeSpecific, 25)  
printRF(seedNum3, treeSpecific)  
printRF(seedNum3, treeSpecific, 5)  
printRF(seedNum3, treeSpecific, 10)  
printRF(seedNum3, treeSpecific, 15)  
printRF(seedNum3, treeSpecific, 25)

What we do see is that as we increase the trees, it does not necessarily perform much better past 10; however, there is definitely the fear of overfitting. We will be adding in 3 and 10 tree variations of the two datasets into our final dataframe.

confusionTableRF <- function(seedNum, dataSet, ntrees=3){  
 # set seed  
 set.seed(seedNum)  
 # Generate random sample of rows  
 randIndex <- sample(1:nrow(dataSet))  
 cutPoint <- floor(nrow(dataSet)\*2/3)  
 train <- dataSet[randIndex[1:cutPoint],]  
 test <- dataSet[randIndex[(cutPoint+1):length(randIndex)],]  
 algorithm <- randomForest(Attrition~., data = train, ntree=ntrees)  
 print("importance")  
 print(importance(algorithm))  
 predicted <- predict(algorithm, test, type="class")  
 set.seed(NULL)  
 return(table(predictedAttrition=predicted, actualAttrition=test$Attrition))  
}  
  
  
completeRFFunc <- function(dataSet, ntrees=3){  
 table1 <- confusionTableRF(seedNum1, dataSet, ntrees)  
 table2 <- confusionTableRF(seedNum2, dataSet, ntrees)  
 table3 <- confusionTableRF(seedNum3, dataSet, ntrees)  
 table4 <- confusionTableRF(seedNum4, dataSet, ntrees)  
 table5 <- confusionTableRF(seedNum5, dataSet, ntrees)  
   
 tableCalc1 <- tableCalc(table1)  
 tableCalc2 <- tableCalc(table2)  
 tableCalc3 <- tableCalc(table3)  
 tableCalc4 <- tableCalc(table4)  
 tableCalc5 <- tableCalc(table5)  
   
 tableCalc <- data.frame(rbind(as.matrix(tableCalc1),as.matrix(tableCalc2),as.matrix(tableCalc3),as.matrix(tableCalc4),as.matrix(tableCalc5)))  
   
 avgTableCalc <- averageTableCalc(tableCalc)  
 print(avgTableCalc)  
}

rfHRTree <- completeRFFunc(HR\_tree, 3)

## [1] "importance"  
## MeanDecreaseGini  
## Age 16.568799  
## BusinessTravel 2.696075  
## DailyRate 12.403092  
## Department 7.567464  
## DistanceFromHome 13.754774  
## Education 4.579637  
## EducationField 10.214950  
## EnvironmentSatisfaction 5.912594  
## Gender 4.442336  
## HourlyRate 9.581976  
## JobInvolvement 6.841059  
## JobLevel 8.990705  
## JobRole 20.140894  
## JobSatisfaction 10.050587  
## MaritalStatus 1.235543  
## MonthlyIncome 17.683327  
## MonthlyRate 16.555428  
## NumCompaniesWorked 10.595557  
## OverTime 7.404556  
## PercentSalaryHike 6.850607  
## PerformanceRating 0.000000  
## RelationshipSatisfaction 5.843753  
## StockOptionLevel 16.441079  
## TotalWorkingYears 14.307302  
## TrainingTimesLastYear 9.069804  
## WorkLifeBalance 6.533326  
## YearsAtCompany 6.351438  
## YearsInCurrentRole 4.529337  
## YearsSinceLastPromotion 11.645180

More Importance Variables Were Cut Out From the Report

## YearsWithCurrManager 10.293177  
## avgAccuracy avgPrecisionYes avgPrecisionNo avgRecallYes avgRecallNo  
## 1 0.8183673 0.3966877 0.8724989 0.2817504 0.9185592

rfHRTree$type <- "rf\_hrTree\_3trees"  
rfTreeSpecific <- completeRFFunc(treeSpecific, 3)

More Importance Variables Were Cut Out From the Report

[results removed for ease of reading]

rfTreeSpecific$type <- "rf\_treeSpecific\_3trees"  
rfHRTree10 <- completeRFFunc(HR\_tree, 10)

[results removed for ease of reading]

rfHRTree10$type <- "rf\_hrTree\_10trees"  
rfTreeSpecific10 <- completeRFFunc(treeSpecific, 10)

[results removed for ease of reading]

rfTreeSpecific10$type <- "rf\_treeSpecific\_10trees"  
completeModels <- rbind(completeModels, rfHRTree, rfHRTree10, rfTreeSpecific, rfTreeSpecific10)

##### KNN

Our last algorithm is K Nearest Neighbor: is a pattern recognition algorithm that trains based on information from the nearest examples. It keeps its training data at all times to use, and thus is fairly computationally heavy.

library(class)  
#convert to numeric  
HR\_factor <- HR\_tree  
HR\_factor$Attrition <-as.numeric(HR\_factor$Attrition)  
HR\_factor$BusinessTravel <- as.numeric(HR\_factor$BusinessTravel)  
HR\_factor$Department <- as.numeric(HR\_factor$Department)  
HR\_factor$Education <- as.numeric(HR\_factor$Education)  
HR\_factor$EducationField <- as.numeric(HR\_factor$EducationField)  
HR\_factor$EnvironmentSatisfaction <- as.numeric(HR\_factor$EnvironmentSatisfaction)  
HR\_factor$Gender <- as.numeric(HR\_factor$Gender)  
HR\_factor$JobInvolvement <- as.numeric(HR\_factor$JobInvolvement)  
HR\_factor$JobLevel <- as.numeric(HR\_factor$JobLevel)  
HR\_factor$JobRole <- as.numeric(HR\_factor$JobRole)  
HR\_factor$JobSatisfaction <- as.numeric(HR\_factor$JobSatisfaction)  
HR\_factor$MaritalStatus <- as.numeric(HR\_factor$MaritalStatus)  
HR\_factor$OverTime <- as.numeric(HR\_factor$OverTime)  
HR\_factor$PerformanceRating <- as.numeric(HR\_factor$PerformanceRating)  
HR\_factor$RelationshipSatisfaction <- as.numeric(HR\_factor$RelationshipSatisfaction)  
HR\_factor$StockOptionLevel <- as.numeric(HR\_factor$StockOptionLevel)  
HR\_factor$WorkLifeBalance <- as.numeric(HR\_factor$WorkLifeBalance)  
  
printNN <- function(seedNum, dataSet, kGuess=3){  
 # set seed  
 set.seed(seedNum)  
 # Generate random sample of rows  
 randIndex <- sample(1:nrow(dataSet))  
 cutPoint <- floor(nrow(dataSet)\*2/3)  
 newTrain <- dataSet[randIndex[1:cutPoint],]  
 newTest <- dataSet[randIndex[(cutPoint+1):length(randIndex)],]  
 testNoLabel <- newTest  
 testNoLabel$Attrion <- NULL  
   
 predicted <- knn(train=newTrain, test=testNoLabel, cl=newTrain$Attrition, k=kGuess, prob=FALSE)  
 print(table(predictedAttrition=predicted, actualAttrition=newTest$Attrition))  
 set.seed(NULL)  
}

printNN(seedNum1, HR\_factor, 3)

## actualAttrition  
## predictedAttrition 1 2  
## 1 373 65  
## 2 40 12

printNN(seedNum1, HR\_factor, 5)

## actualAttrition  
## predictedAttrition 1 2  
## 1 394 67  
## 2 19 10

printNN(seedNum2, HR\_factor, 3)

## actualAttrition  
## predictedAttrition 1 2  
## 1 389 57  
## 2 32 12

printNN(seedNum2, HR\_factor, 5)

## actualAttrition  
## predictedAttrition 1 2  
## 1 404 62  
## 2 17 7

printNN(seedNum3, HR\_factor, 3)

## actualAttrition  
## predictedAttrition 1 2  
## 1 391 67  
## 2 18 14

printNN(seedNum3, HR\_factor, 5)

## actualAttrition  
## predictedAttrition 1 2  
## 1 394 71  
## 2 15 10

We can see that on the full dataset, its performance is fairly poor outside of accuracy. It predicts yes to attrition fairly rarely but not with high recall or precision.

factorSpecific <- data.frame("Attrition"=HR\_factor$Attrition, "BusinessTravel"=HR\_factor$BusinessTravel, "Department"=HR\_factor$Department, "Education"=HR\_factor$Education, "JobLevel"=HR\_factor$JobLevel, "MaritalStatus"=HR\_factor$MaritalStatus, "Overtime"=HR\_factor$OverTime, "WorkLifeBalance"=HR\_factor$WorkLifeBalance, "YearsInCurrentRole"=HR\_factor$YearsInCurrentRole, "YearsWithCurrManager"=HR\_factor$YearsWithCurrManager )  
printNN(seedNum1, factorSpecific, 3)

## actualAttrition  
## predictedAttrition 1 2  
## 1 412 37  
## 2 1 40

printNN(seedNum1, factorSpecific, 5)

[results removed for ease of reading]

printNN(seedNum2, factorSpecific, 3)

[results removed for ease of reading]

printNN(seedNum2, factorSpecific, 5)

[results removed for ease of reading]

printNN(seedNum3, factorSpecific, 3)

[results removed for ease of reading]

printNN(seedNum3, factorSpecific, 5)

[results removed for ease of reading]

With the specific 10, it’s accuracy is fairly high on accuracy and exceptional on yes recall. We will add the specific 10 values with both 3 and 10 nearest neighbors into our final dataframe.

confusionTableNN <- function(seedNum, dataSet, kGuess=3){  
 # set seed  
 set.seed(seedNum)  
 # Generate random sample of rows  
 randIndex <- sample(1:nrow(dataSet))  
 newTrain <- dataSet[randIndex[1:cutPoint],]  
 newTest <- dataSet[randIndex[(cutPoint+1):length(randIndex)],]  
 testNoLabel <- newTest  
 testNoLabel$Attrion <- NULL  
   
 predicted <- knn(train=newTrain, test=testNoLabel, cl=newTrain$Attrition, k=kGuess, prob=FALSE)  
 set.seed(NULL)  
 return(table(predictedAttrition=predicted, actualAttrition=newTest$Attrition))  
}  
  
tableCalc2 <- function(newTable){  
 calcTable <- as.data.frame(as.matrix.data.frame(newTable))  
 accurateNumbers <- 0  
 totalNumbers <- 0  
 precision <- data.frame()  
 recall <- data.frame()  
 for(i in 1:length(calcTable)){  
 columnSum <- sum(calcTable[,i])  
 rowSum <- sum(calcTable[i,])  
 cell <- calcTable[i,i]  
 accurateNumbers <- accurateNumbers + cell  
 totalNumbers <- totalNumbers + columnSum  
 precision[1,i] <- cell / columnSum  
 recall[1,i] <- cell / rowSum  
 }  
 dataFrame <- data.frame("precisionNo"=precision[1,1], "precisionYes"=precision[1,2], "recallNo"=recall[1,1],"recallYes"=recall[1,2], "accuracy"=accurateNumbers/totalNumbers)  
}  
  
averageTableCalc2 <- function(dataFrame){  
 avgAccuracy <- mean(dataFrame$accuracy)  
 avgPrecisionYes <- mean(dataFrame$precisionYes)  
 avgPrecisionNo <- mean(dataFrame$precisionNo)  
 avgRecallYes <- mean(dataFrame$recallYes)  
 avgRecallNo <- mean(dataFrame$recallNo)  
 newDF <- data.frame(avgAccuracy, avgPrecisionYes, avgPrecisionNo, avgRecallYes, avgRecallNo)  
 return(newDF)  
}  
  
  
completeNNFunc <- function(dataSet, kGuess=3){  
 table1 <- confusionTableNN(seedNum1, dataSet, kGuess)  
 table2 <- confusionTableNN(seedNum2, dataSet, kGuess)  
 table3 <- confusionTableNN(seedNum3, dataSet, kGuess)  
 table4 <- confusionTableNN(seedNum4, dataSet, kGuess)  
 table5 <- confusionTableNN(seedNum5, dataSet, kGuess)  
   
 tableCalc1 <- tableCalc2(table1)  
 tableCalc2 <- tableCalc2(table2)  
 tableCalc3 <- tableCalc2(table3)  
 tableCalc4 <- tableCalc2(table4)  
 tableCalc5 <- tableCalc2(table5)  
   
 tableCalc <- data.frame(rbind(as.matrix(tableCalc1),as.matrix(tableCalc2),as.matrix(tableCalc3),as.matrix(tableCalc4),as.matrix(tableCalc5)))  
   
 avgTableCalc <- averageTableCalc2(tableCalc)  
 print(avgTableCalc)  
}

nn3 <- completeNNFunc(factorSpecific, 3)

[results removed for ease of reading]

nn3$type <- "nn\_treeSpecific\_3"  
nn10 <- completeNNFunc(factorSpecific, 10)

[results removed for ease of reading]

nn10$type <- "nn\_treeSpecific\_10"  
completeModels <- rbind(completeModels, nn3, nn10)  
completeModels

[results removed for ease of reading]

completeModels <- subset(completeModels, select=c(6,1:5))

formattable(completeModels, align = c("l",rep("r", NCOL("type") - 1)), list(  
 `type` = formatter("span", style = ~ style(color = "#000000",font.weight = "bold")),   
 area(col = 2:6) ~ color\_tile("#ff0000", "#71CA97")))



## Results

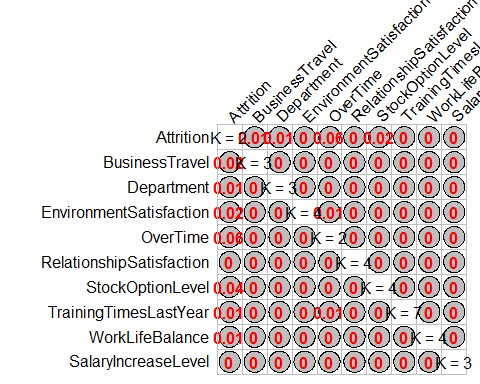
### Exploratory Data Analysis & Visualization

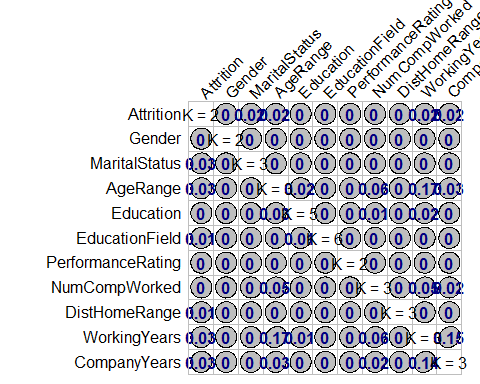
Exploratory Data Analysis and Visualization showed that there was not strong association between any one attribute and attrition. The Goodman and Kruskal Tau measure model was used to establish association of categorical values. To make these associations easier to visualize, we grouped the attributes in 3 groups:

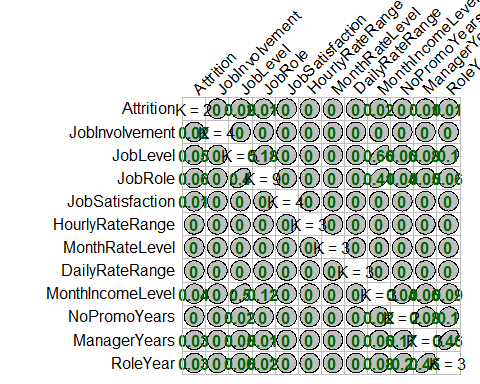
Person/Profile Company Role/Job

and compared them to Attrition.

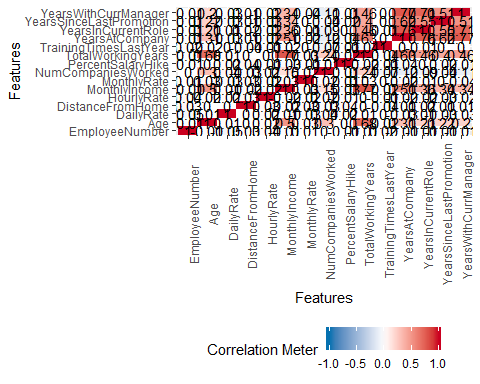
Each group showed low association to attrition.







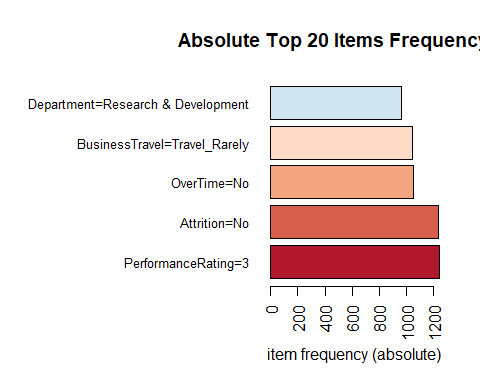
Attributes were correlated to each other, and we can see pockets of correlation between attributes. We can use this information to simplify models in later stages.



What was interesting and unexpected was that the attribute to attribute correlation chart showed actionable information that could be used for simplifying models while the direct correlation chart was relatively inconclusive.

Additionally, each attribute was correlated to attrition individually. The findings confirmed that there was any singular or group of attributes that could be strongly correlated to attrition and further work with more advanced techniques should be used to identify important attributes.

### Association Rule Mining



Conversion of the data to transacions allowed for an initial assessment of the most frequent responses. Attrition, the attribute of interest in this study had 83.9% “No” responses (1233 out of the 1470 transactions), followed by Overtime with 71.7% “No” responses and Business Travel with 70.9% “Travel Rarely” responses. Considering other data analytics indicated Frequent business travel and overtime as a driver for attrition, knowing that only 30% or less of the respondants had those responses is key information when it comes to deciding what type of startegies to implement and who shoudld be the target audience.

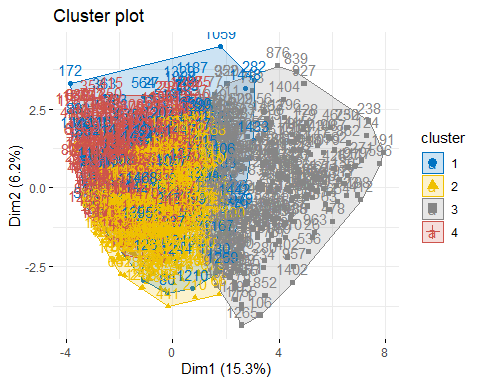
By fixing the RHS to Attrition = Yes and Attrition = No rules provide more insight.

With Attrition = Yes, the most frequent factors in the top 20 rules are: \* Marital Status = Single. In 13 out of the 20 rules \* Overtime = Yes. In 18 out of the 20 rules \* Years with current Manager = 0. In 16 out of the 20 rules \* Years in current Role = 0. In 12 out of the 20 rules \* Low Income. In 10 out of the 20 rules

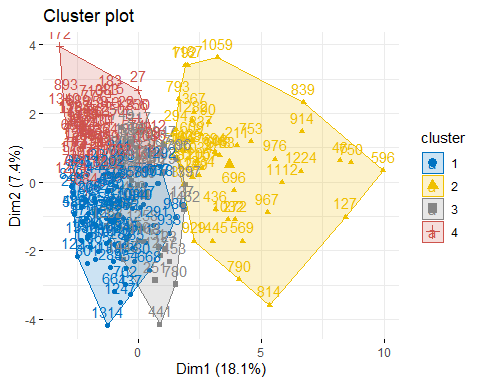
With Attrition = No, the most frequent factors in the top 20 rules are: \* Department=Research & Development. In 10 out of the 20 rules  
\* OverTime=No. In 15 out of the 20 rules  
\* StockOptionLevel=1. In 6 out of the 20 rules  
\* WorkLifeBalance=3. In 11 out of the 20 rules

### K-means Clustering

K-means was run with k=2, 3, 4, 5, and 6 clusters, both with and without the attribution attribute. The most interesting cluster was k=4, scaled data, with attribution included. This shows one group that separates clearly, and three that overlap.



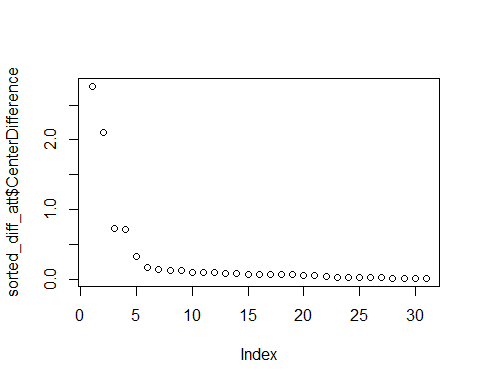
When looking at only the people who left (attrition = yes), notice how few people left in the right group (cluster 1).



The 4-cluster version shows clear separation on several attributes between the group with the highest attrition and the one with the lowest attrition.



Because of sizing, all attributes are not shown on the Y-Axis.

 The ten most influential attributes were:

sorted\_diff\_att[1:11, ]

## X1 X2 CenterDifference  
## PerformanceRating 2.34535291 -0.42608501 2.77  
## PercentSalaryHike 1.79601539 -0.31523886 2.11  
## MaritalStatus 0.02586244 -0.70216265 0.73  
## StockOptionLevel 0.01074170 0.72037503 0.71  
## Attrition 0.06971126 -0.26053141 0.33  
## Education -0.09743924 0.05873249 0.16  
## EnvironmentSatisfaction -0.08737319 0.05573561 0.14  
## JobInvolvement -0.03922610 0.08225515 0.12  
## NumCompaniesWorked -0.07329688 0.04583451 0.12  
## DailyRate -0.01343231 0.07443559 0.09  
## YearsInCurrentRole -0.10230522 -0.19278390 0.09

### Predictive Models



When looking at the complete table, we do need to define what is our success criteria for defining how well a model performs.

We can look at accuracy, precisionYes, precisionNo, recallYes, and recallNo and decide across a combination of metrics to best define what makes the most sense.

Because we ultimately want to maximize employees who are likely to leave, we should weight Yes.

When we look at recallYes, which provides us with insight on the percentage of correctly classified relevant results, K Nearest Neighbors immediately handles best followed by Naive Bayes.

The best Accuracy is KNN followed by SVM.

The best on precisionYes is SVM followed by RF with 10 trees on the complete dataset.

Each of these has its cons. KNN has fairly low precision on Yes meaning that only when it’s certain, will it make a move. And that certainty is between 30-47% of the of time. But when it does count, the data is indredibly accurate.

SVM was high on PrecisionYes but medium on Recall. That means that it modeled more employees as being likely to leave, but of those, only 49% were truly likely to leave.

Ultimately, this is a question of is it better to classify someone as leaving when they’re staying or to classify someone as staying when they’re leaving?

KNN, SVM, and Naive Bayes all give some distinct information. However, Random Forest can do fairly well and provides a lot of output that we can follow clearly.

## Conclusions

### Next steps in analysis

Analysis is best done iteratively. To further improve on the models, it is recommended that future analysis include these steps.

*Gather more data* \* More data leads to better results. It would be better to have several thousand observations. \* Collect more attributes. Research has shown that some other factors that weren’t studied here can impact attrition, including onboarding experience and the networking of employees. \* Get a balanced sample. Some models work better when the “yes” and “no” classes have similar numbers of observations. \* Compare the models’ predictions with actual attrition to see what parameters they may have chosen to be groups.

*Focus on the most successful models* All models had some good qualities. We recommend continuing with:

* KNN (very accurate at identifying who will quit but not as useful for large data, and may need to have cutoffs based on timeframes)
* SVM (very little processing work, scales well but doesn’t really provide insight into what business variables to improve)
* Random forest (good at handling a variety of attributes, provides attributes, but is less accurate)
* Naive Bayes (all rounder, but doesn’t excel at anything in particular)

*Run the models on the new data every quarter*

* This helps to identify if there are changes at the company, and whether new programs to influence attrition are working.

### Business decisions

In addition to using the models to predict if an individual employee is going to leave, the models also identified common contributions to attrition. The HR department can develop programs to target these factors.

Of the attributes that the models chose as influential, the most common were:

* Overtime (in 5 models)
* Environmental satisfaction (4)
* Job level (4)
* Marital status (4)
* Monthly income (4)
* Work-life balance (4)

The company cannot impact marital status, and it is illegal to hire based on that attribute. However, the company can influence the others. For example, it could reduce overtime…or pay people who work overtime more money. There are lots of possible ways of addressing these issues, and the HR department should look further at things like environmental satisfaction and work-life balance by interviewing at-risk employees.

## References

[1] “Why People Quit Their Jobs.” Harvard Business Review, Sept. 2016, hbr.org/2016/09/why-people-quit-their-jobs. Accessed 10 Mar. 2020.

[2] “Why People Quit Their Jobs.” Harvard Business Review, Sept. 2016, hbr.org/2016/09/why-people-quit-their-jobs. Accessed 10 Mar. 2020.

[3] “How to Predict Turnover on Your Sales Team.” Harvard Business Review, July 2017, hbr.org/2017/07/how-to-predict-turnover-on-your-sales-team. Accessed 10 Mar. 2020.

[4] “How to Predict Turnover on Your Sales Team.” Harvard Business Review, July 2017, hbr.org/2017/07/how-to-predict-turnover-on-your-sales-team. Accessed 10 Mar. 2020.

[5] “To Retain New Hires, Make Sure You Meet with Them in Their First Week.” Harvard Business Review, 14 June 2018, hbr.org/2018/06/to-retain-new-hires-make-sure-you-meet-with-them-in-their-first-week. Accessed 10 Mar. 2020.

[6] “How to Predict Turnover on Your Sales Team.” Harvard Business Review, July 2017, hbr.org/2017/07/how-to-predict-turnover-on-your-sales-team. Accessed 10 Mar. 2020.

[7] “8 Things Leaders Do That Make Employees Quit.” Harvard Business Review, 10 Sept. 2019, hbr.org/2019/09/8-things-leaders-do-that-make-employees-quit. Accessed 10 Mar. 2020.

[8] “Work Institute Releases National Employee Retention Report.” Businesswire.Com, May 2018, www.businesswire.com/news/home/20180501006594/en/Work-Institute-Releases-National-Employee-Retention-Report. Accessed 10 Mar. 2020.

[9] “How to Predict Turnover on Your Sales Team.” Harvard Business Review, July 2017, hbr.org/2017/07/how-to-predict-turnover-on-your-sales-team. Accessed 10 Mar. 2020.

[10] Maurer, Roy. “Onboarding Key to Retaining, Engaging Talent.” SHRM, SHRM, 16 Apr. 2015, www.shrm.org/ResourcesAndTools/hr-topics/talent-acquisition/Pages/Onboarding-Key-Retaining-Engaging-Talent.aspx. Accessed 10 Mar. 2020.

[11] “The Battle Against Executive Attrition.” Harvard Business Review, 17 July 2008, hbr.org/2008/07/the-battle-against-executive-a. Accessed 10 Mar. 2020.

[12] Dowsett, C. (2018, April). It’s Time to Talk About Organizational Bias in Data Use. Medium; Towards Data Science.