Problem statement

**Students are given the data file collected by the HR department.**

Build Neural Network and CART model on same:

HR Employee Attrition Data file is provided:

1) HR\_Employee\_Attrition\_Data.csv

2) HR\_Employee\_Attrition\_Metadata.xlsx

**Steps involved should be:**

**Build Neural Network and CART model on same:**  
**Steps involved should be:**

**a) Data Import (Target variable is "Attrition" column)  
b) Split the data in Dev & Hold Out sample (70:30)  
c) Perform Exploratory Data Analysis  
d) Identify columns which are of no use. drop those columns  
e) Build Neural Network Model (Development sample)  
f) Validate the NN model on Hold Out. If need be improvise  
g) Build CART Model  
h) Validate CART Model  
i) Compare NN with CART  
j) Check whether NN Model Performance outperforms the CART Model Performance**

**INTRODUCTION**

This data set presents an employee survey, indicating if there is attrition or not. The data

set contains approximately 3000 entries. Given the limited size of the data set, the model

should only be expected to provide modest improvement in identification of attrition vs a

random allocation of probability of attrition.

We are proposing the below Hypothesis:

* **Null Hypothesis**: Employee attrition is not dependent on any of

the predictor variables

* **Alternative Hypothesis**: Employee attrition is dependent on any

of the predictor variables

**Step 1 ->**Data Import:

**DATA LOAD AND CHECK (IMPORT, CLEANING, SPLIT, SANITY):**

**Step 1: Setting Path / Reading Data**

library(rpart)

library(rpart.plot)

library(randomForest)

library(corrplot)

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

## Attaching package: 'ggplot2'

library(neuralnet)

data = read.csv(file.choose())

**Step 2: Basic Data Check**

Check for Missing Data in provided file

**sum(is.na(data))**

## [1] 0

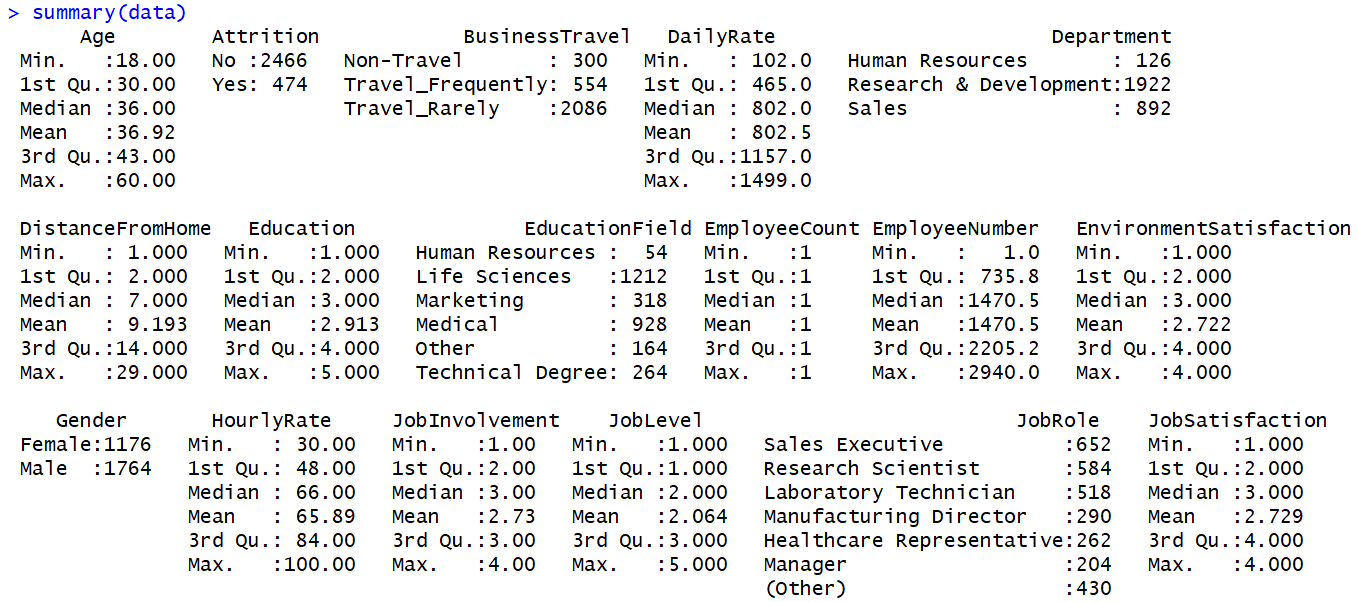
If there were any missing values, then can locate missing values using colSums(is.na(data).

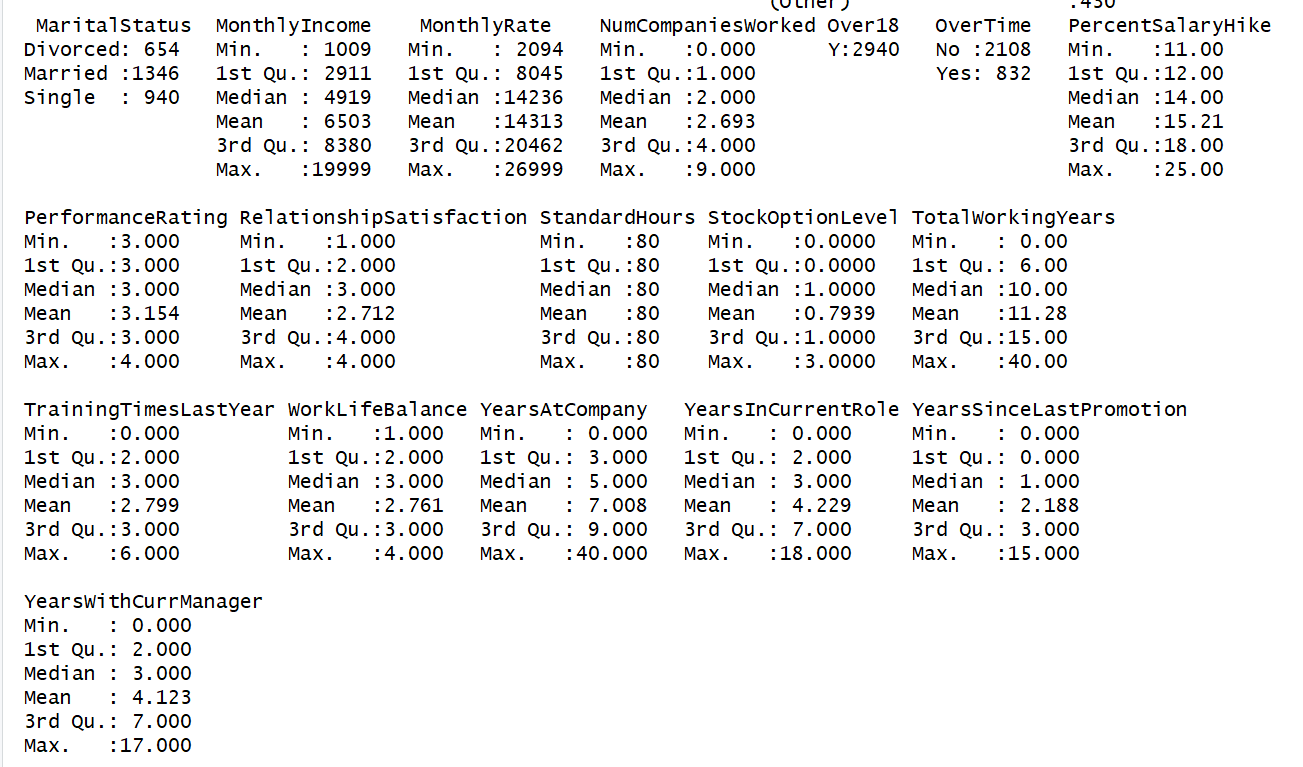
Summary of data, identifying data types, min-max values. Also, need to check for likert scale

data, if any incorrect values are present, such as Values out of range of scale, or decimal

values

summary(data)





Minimum, Maximum, Mean, First Quartile, Median and Third Quartile values of

the following variables are same indicating the constant value they hold.

EmployeeCount having constant value 1

StandardHours having constant value 80

Over18 variable has constant value Y for all the values for all 2940 observations.

EmployeeNumber is an identifier variable and there are 2940 values for this

variable matching with the total number of observations.

Identify columns which are of no use. drop those columns

a) EmployeeCount

b) StandardHours

c) Over18

d) EmployeeNumber

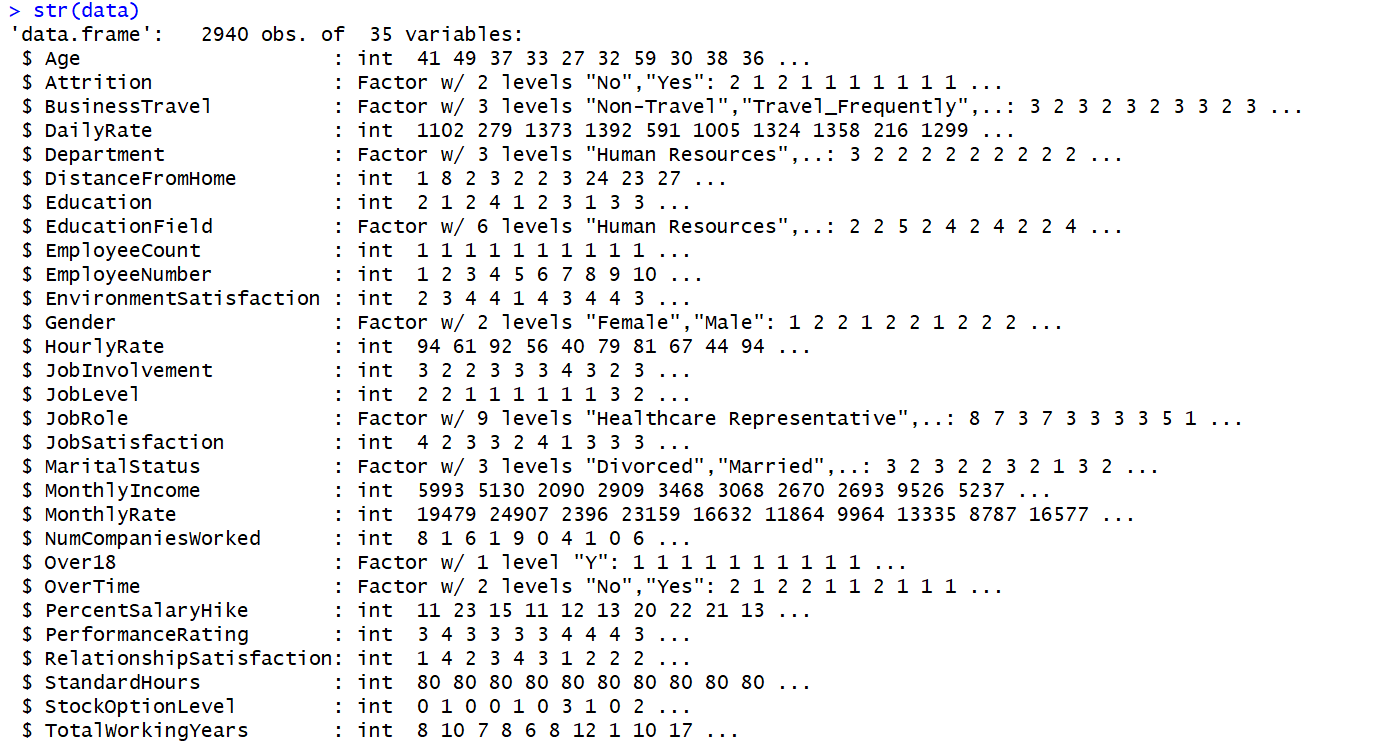
>temp = read.csv('HR\_Employee\_Attrition\_Data.csv', header = FALSE)

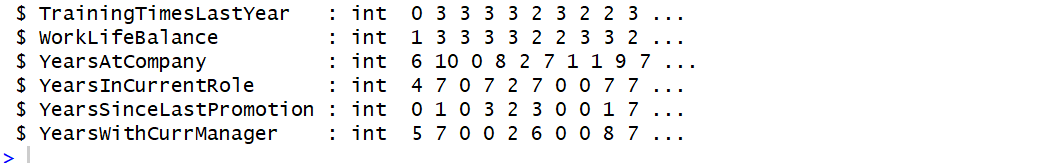
> summary(temp)

>rm(temp)

No anomaly seen in the data. Summary shows some variables have only single value or

unimportant value, these will be removed later when data is split.





**Step 3: Data Split - Train & Test**

We split the data into an estimation sample and a validation/test sample - using a

randomized splitting technique. The second validation/test data mimic out-of-sample data,

and the performance on this validation set is a better approximation of the performance

one should expect in practice from the selected classification method.

The split used is 70% estimation, and 30% test data, depending on the number of observations - for example,

when there is a lot of data, you may only keep a few hundreds of them for the validation and

test sets, and use the rest for estimation.

Random sampling 70:30, by generating 70% of total values Train-Data and Test-Data will

be generated based on random rows

set.seed(123)

> s = sample(1:nrow(data), size = 0.7\*nrow(data))

> s = sample(1:nrow(data), size = 0.7\*nrow(data))

> trainD = data[s,]

> testD = data[-s,]

> rm(s)

Calculating ratio of “Yes” targets (attrite employees) in all 3 samples (Data, Train, Test)

Attrition rate in all 3 samples are similar. Random sampling has divided

data in train and test, which are similar in nature

>sum(ifelse(data$Attrition=="Yes",1,0))/nrow(data)

[1] 0.1612245

>sum(ifelse(trainD$Attrition=="Yes",1,0))/nrow(trainD)

[1] 0.1608358

>sum(ifelse(testD$Attrition=="Yes",1,0))/nrow(testD)

[1] 0.1621315

**Step 4: Cleaning & Sanity Check**

Further analysis will be done on Train data, while any cleaning etc. done on Train data will

be copied to Test data as well

Summary of data had shown some unimportant variables: Employee count always 1,

Remove Employee number, Remove Over 18 is always 1, Remove Standard

Hrs is always 80.

> trainD = trainD[,-c(9,10,22,27)]

> names(trainD)

[1] "Age" "Attrition" "BusinessTravel" "DailyRate"

[5] "Department" "DistanceFromHome" "Education" "EducationField"

[9] "EnvironmentSatisfaction" "Gender" "HourlyRate" "JobInvolvement"

[13] "JobLevel" "JobRole" "JobSatisfaction" "MaritalStatus"

[17] "MonthlyIncome" "MonthlyRate" "NumCompaniesWorked" "OverTime"

[21] "PercentSalaryHike" "PerformanceRating" "RelationshipSatisfaction" "StockOptionLevel"

[25] "TotalWorkingYears" "TrainingTimesLastYear" "WorkLifeBalance" "YearsAtCompany"

[29] "YearsInCurrentRole" "YearsSinceLastPromotion" "YearsWithCurrManager"

**Step 4: Rearranging Data**

> trainD= trainD[, c(2,1,4,6,11,17,18,19,21,25,26,28,29,30,31,7,9,12,13,15,22,23,24,27,3,20,5,8,10,14,16)]

**Step 5: Creating new variables**

‘AvgYrs’ : Average years spent in a company (including current company)

‘delta’ : Extra years spent in current company as compared to average years in a company

> trainD$AvgYrs = round(trainD$TotalWorkingYears/(trainD$NumCompaniesWorked+1),1)

> trainD$delta = trainD$YearsAtCompany - trainD$AvgYrs

**EXPLORATORY ANALYSIS**

**Correlation Plot**

Removing target variable ‘Attrition’ and nominal variables from Training Sample, creating

new Sample ‘trainE’ Correlation matrix generated on ‘trainE’ and Correlation plotted to

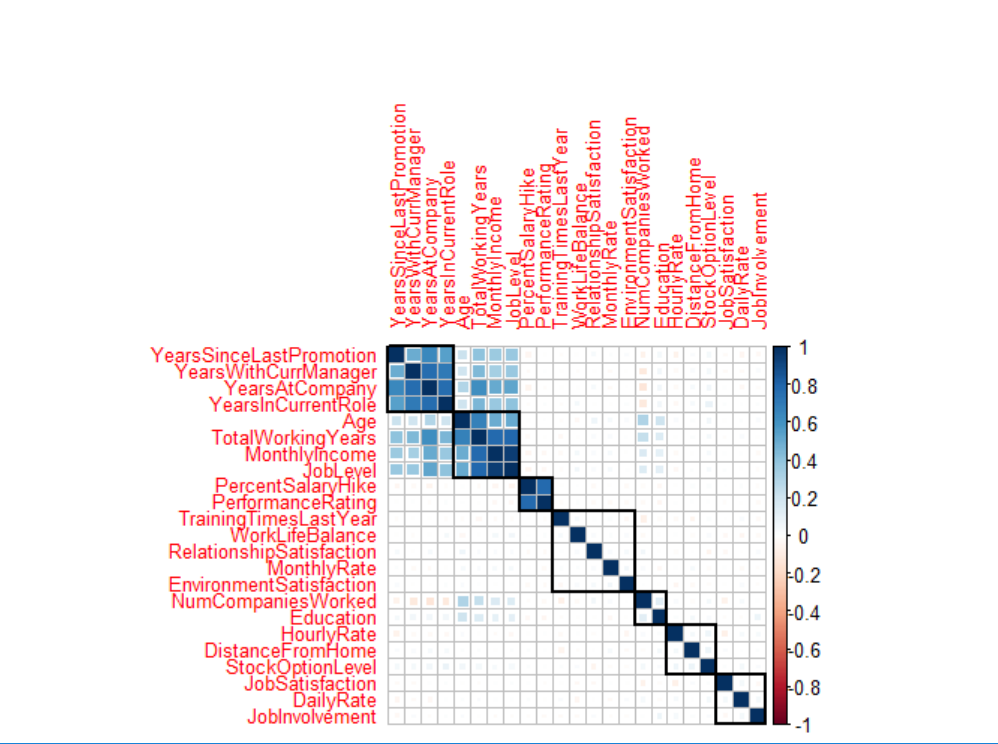
understand any relation between independent variables.

> trainE = trainD[,-c(1, 25:33)]

> trainCor = cor(trainE)

> corrplot(trainCor, method = "square", type = "full", order = "hclust",

+ hclust.method = 'ward.D2', addrect = 7, tl.cex = .8)



> rm(trainE)

> rm(trainCor)

Correlation Matrix suggests few relations:

* Performance rating ~ Percent Salary Hike
* Job Level, Monthly Income, Total working years, Age
* All Years related to current company

These correlation are logical, and may lead to dimension reduction later.

**Attrition:**

> table(data$Attrition)

No Yes

2466 474

> ##

> ## No Yes

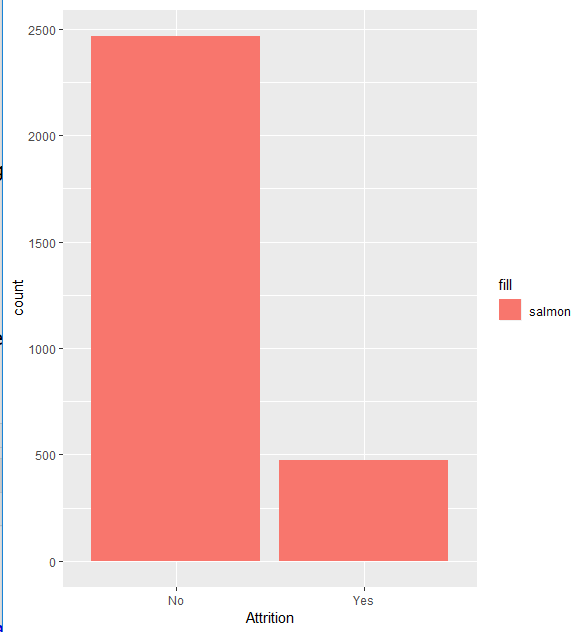
> ## 2466 474

> prop.table(table(data$Attrition))

No Yes

0.8387755 0.1612245

ggplot(data, aes(Attrition, fill="salmon")) + geom\_bar()



16%. 474 out of 2940 have attrited.

As per the CART model, Overtime, MonthlyIncome, TotalWorkingYears, HourlyRate, JobRole and Age are the most important factors influencing the attrition rates. Let’s explore these variables.

**Overtime**

> prop.table(table(data$OverTime))

No Yes

0.7170068 0.2829932

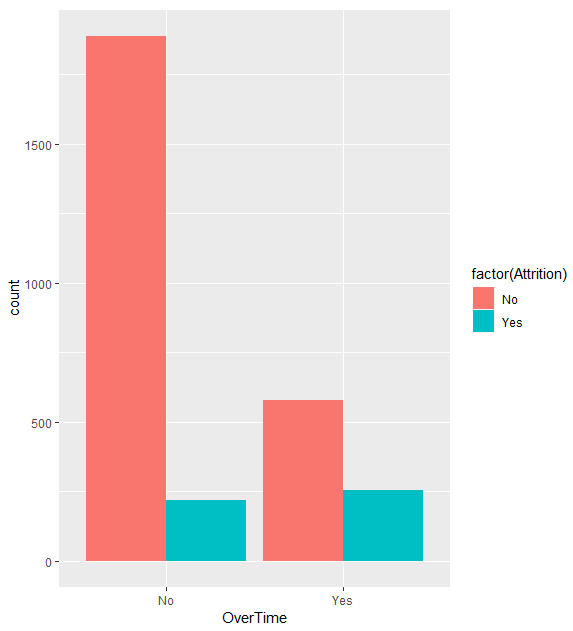
> table(data$OverTime, data$Attrition)

No Yes

No 1888 220

Yes 578 254

> ggplot(data, aes(OverTime, ..count.., fill = factor(Attrition))) + geom\_bar(position="dodge")



Overall 28% of the employees are putting overtime. The percentage of attrition amongst those putting in overtime is close to 44% (254/578) vs 11% (220/1888) for those not putting in overtime. Thus Overtime is contributing towards attrition.

**Monthly Income:**

> summary(data$MonthlyIncome)

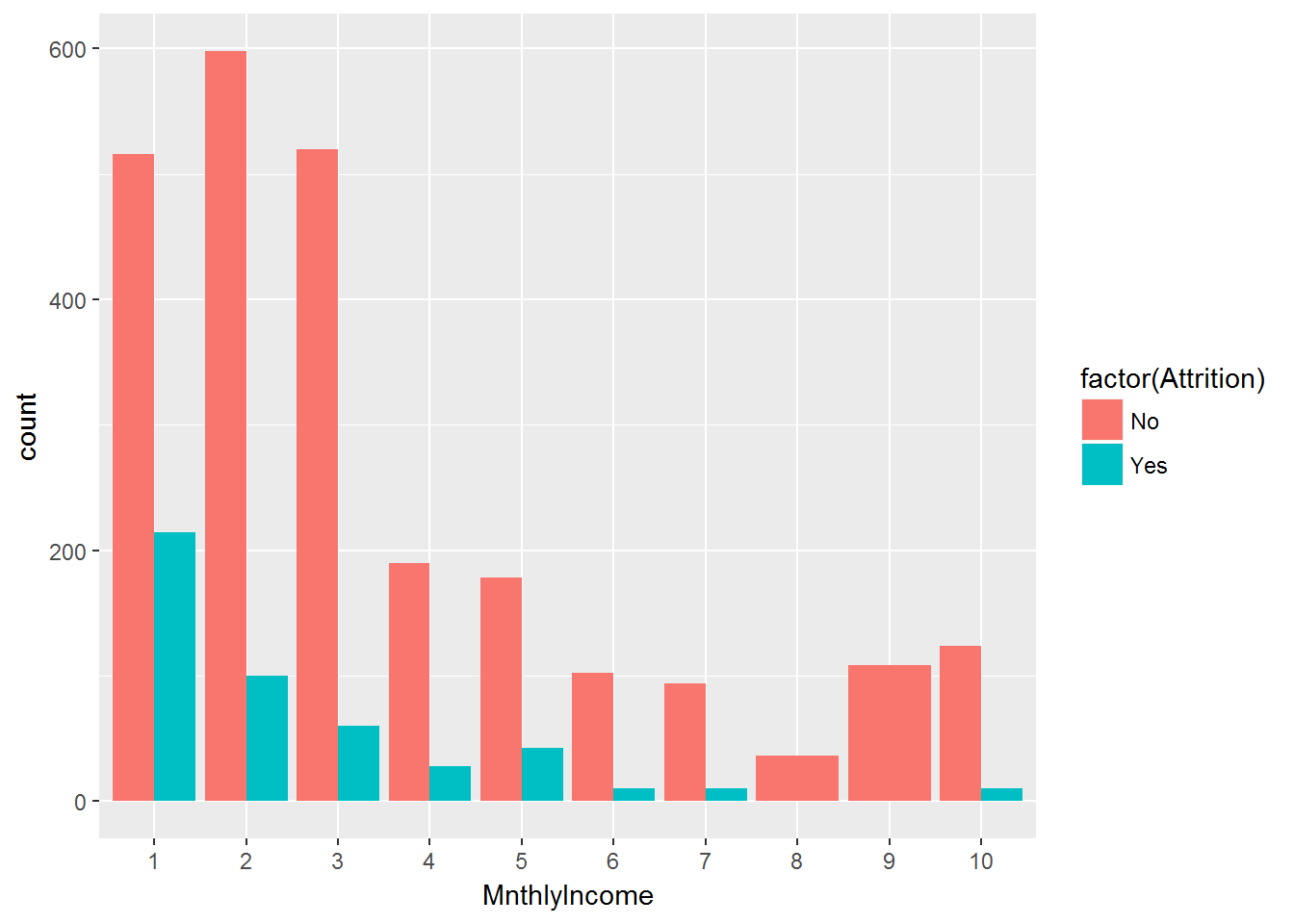
Min. 1st Qu. Median Mean 3rd Qu. Max.

Gb 1009 2911 4919 6503 8380 19999

>

> MnthlyIncome <- cut(data$MonthlyIncome, 10, include.lowest = TRUE, labels=c(1,2,3,4,5,6,7,8,9,10))

> ggplot(data, aes(MnthlyIncome, ..count.., fill = factor(Attrition))) + geom\_bar(position="dodge")

****

The attrition in absolute terms decreases as the salary increases, thus lower salary is contributing towards attrition.

**Total Working Hours:**

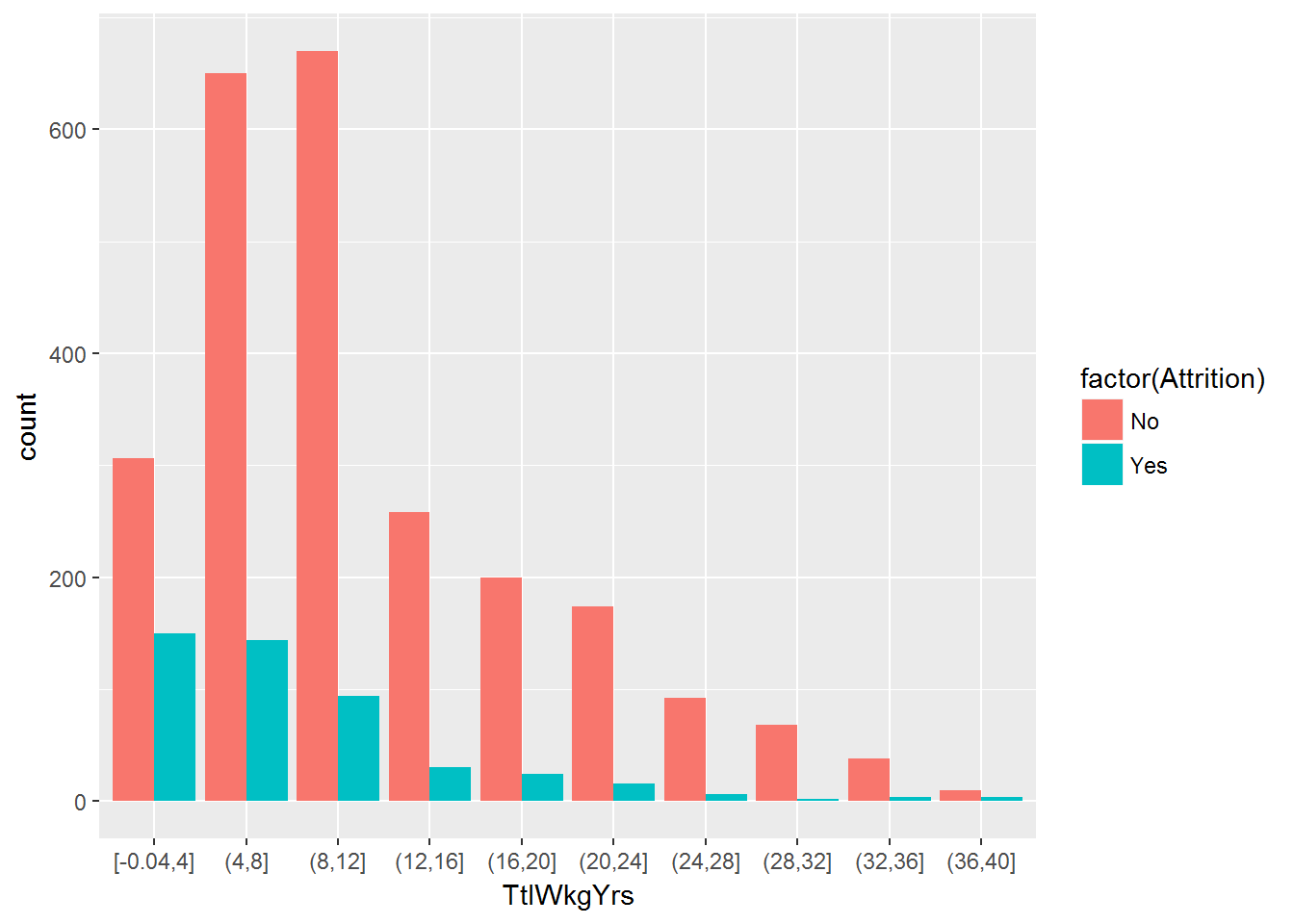
> summary(data$TotalWorkingYears)

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.00 6.00 10.00 11.28 15.00 40.00

> TtlWkgYrs <- cut(data$TotalWorkingYears, 10, include.lowest = TRUE)

> ggplot(data, aes(TtlWkgYrs, ..count.., fill = factor(Attrition))) + geom\_bar(position="dodge")

**** The attrition in absolute terms decreases as the total number of working years increase. After an employee has spent 8-12 years in the company, his chances of attrition decrease.

**Hourly Rate:**

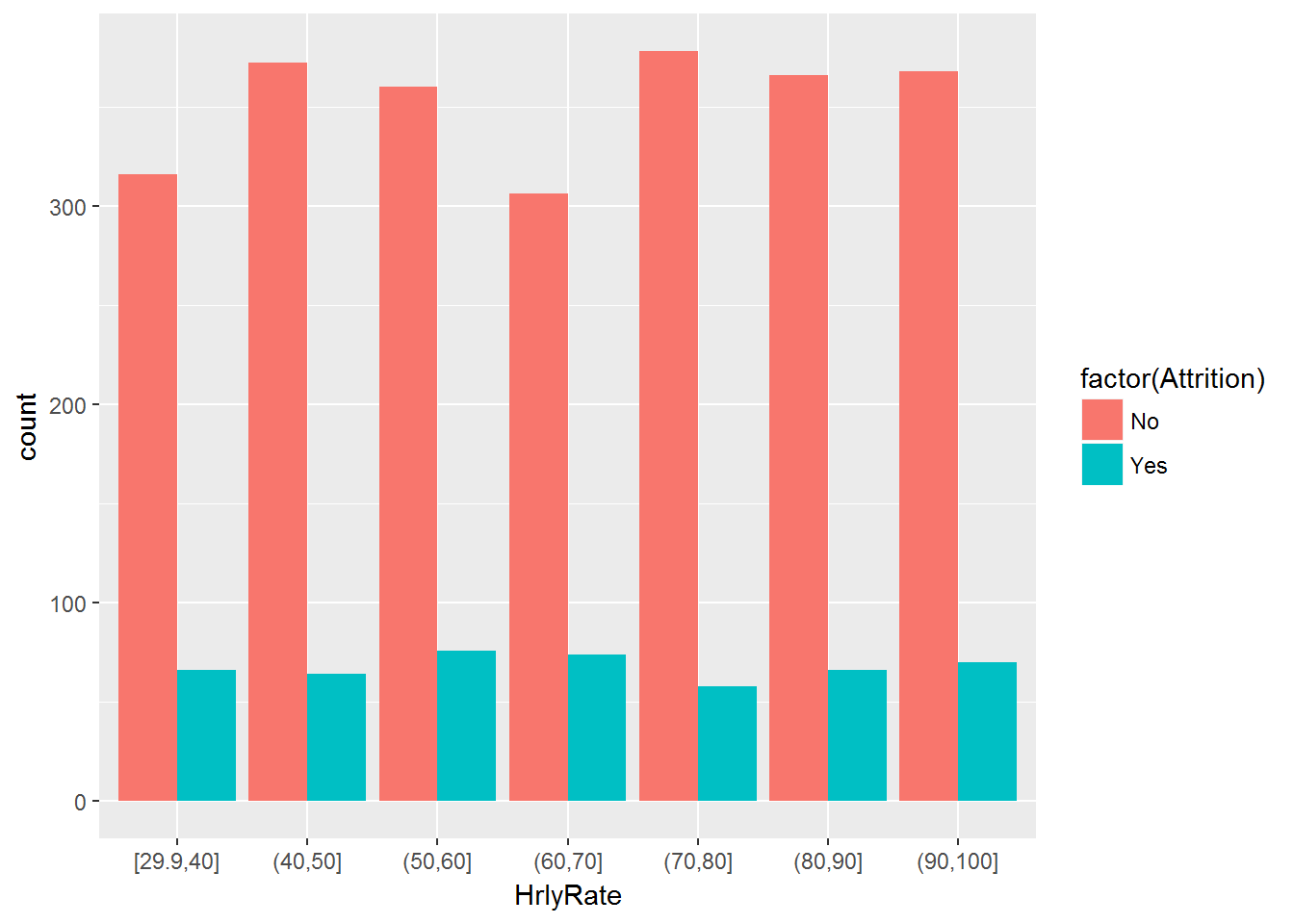
> summary(data$HourlyRate)

Min. 1st Qu. Median Mean 3rd Qu. Max.

30.00 48.00 66.00 65.89 84.00 100.00

> HrlyRate<- cut(data$HourlyRate, 7, include.lowest = TRUE)

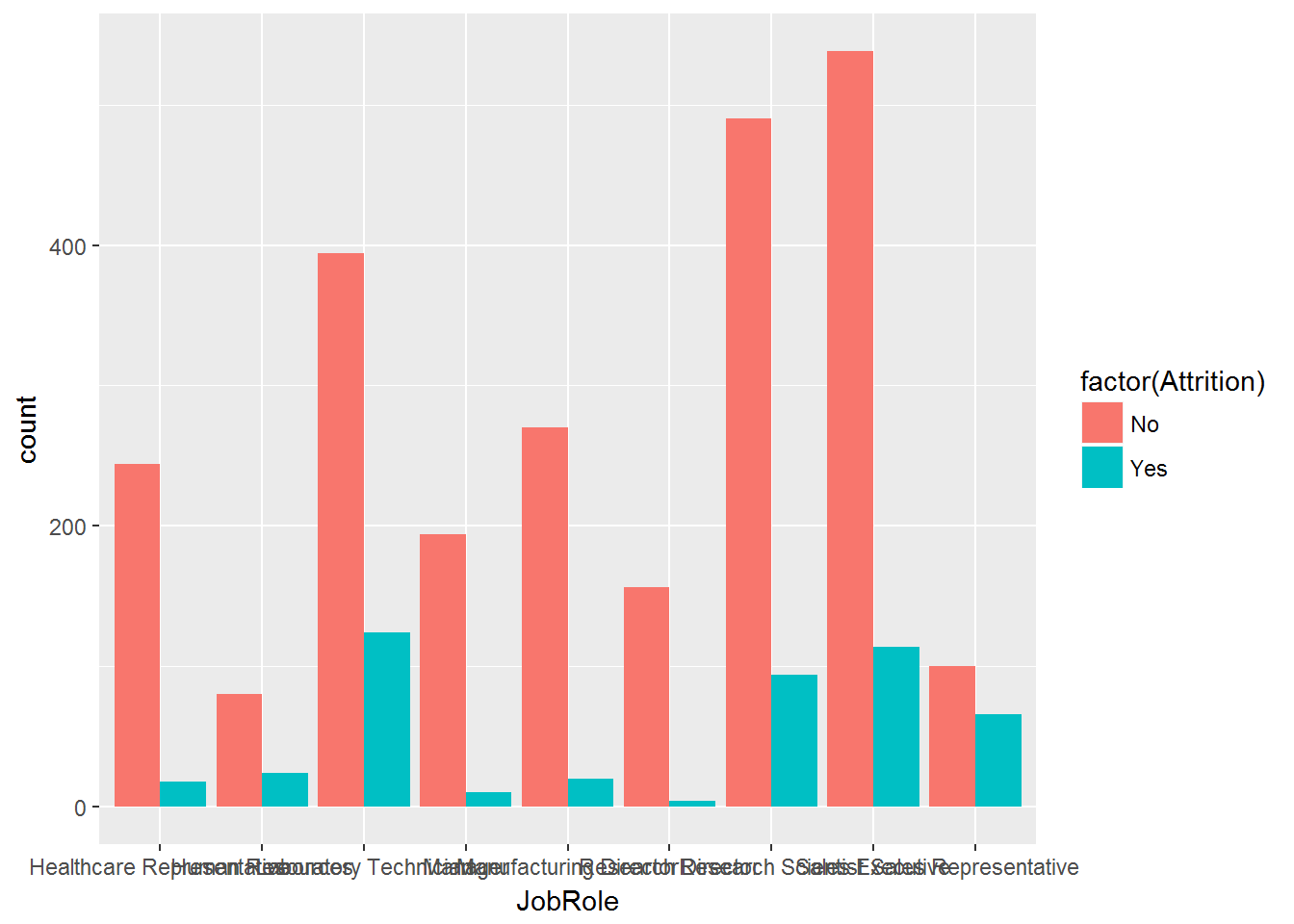
> ggplot(data, aes(HrlyRate, ..count.., fill = factor(Attrition))) + geom\_bar(position="dodge")

****

 There is no pattern in attrition that can be seen in the context of hourly rates.

**Job Role:**

|  |
| --- |
| > table(data$JobRole)  Healthcare Representative Human Resources Laboratory Technician Manager  262 104 518 204  Manufacturing Director Research Director Research Scientist Sales Executive  290 160 584 652  Sales Representative  166  > table(data$JobRole, data$Attrition)  No Yes  Healthcare Representative 244 18  Human Resources 80 24  Laboratory Technician 394 124  Manager 194 10  Manufacturing Director 270 20  Research Director 156 4  Research Scientist 490 94  Sales Executive 538 114  Sales Representative 100 66  > ggplot(data, aes(JobRole, ..count.., fill = factor(Attrition))) + geom\_bar(position="dodge") |
|  |
| |  | | --- | | > | |

**** In absolute terms, Laboratory Technicians followed by the Sales Executives are contributing the maximum towards attrition. In percentage terms, Sales Representative are far ahead at 66%.

**Age:**

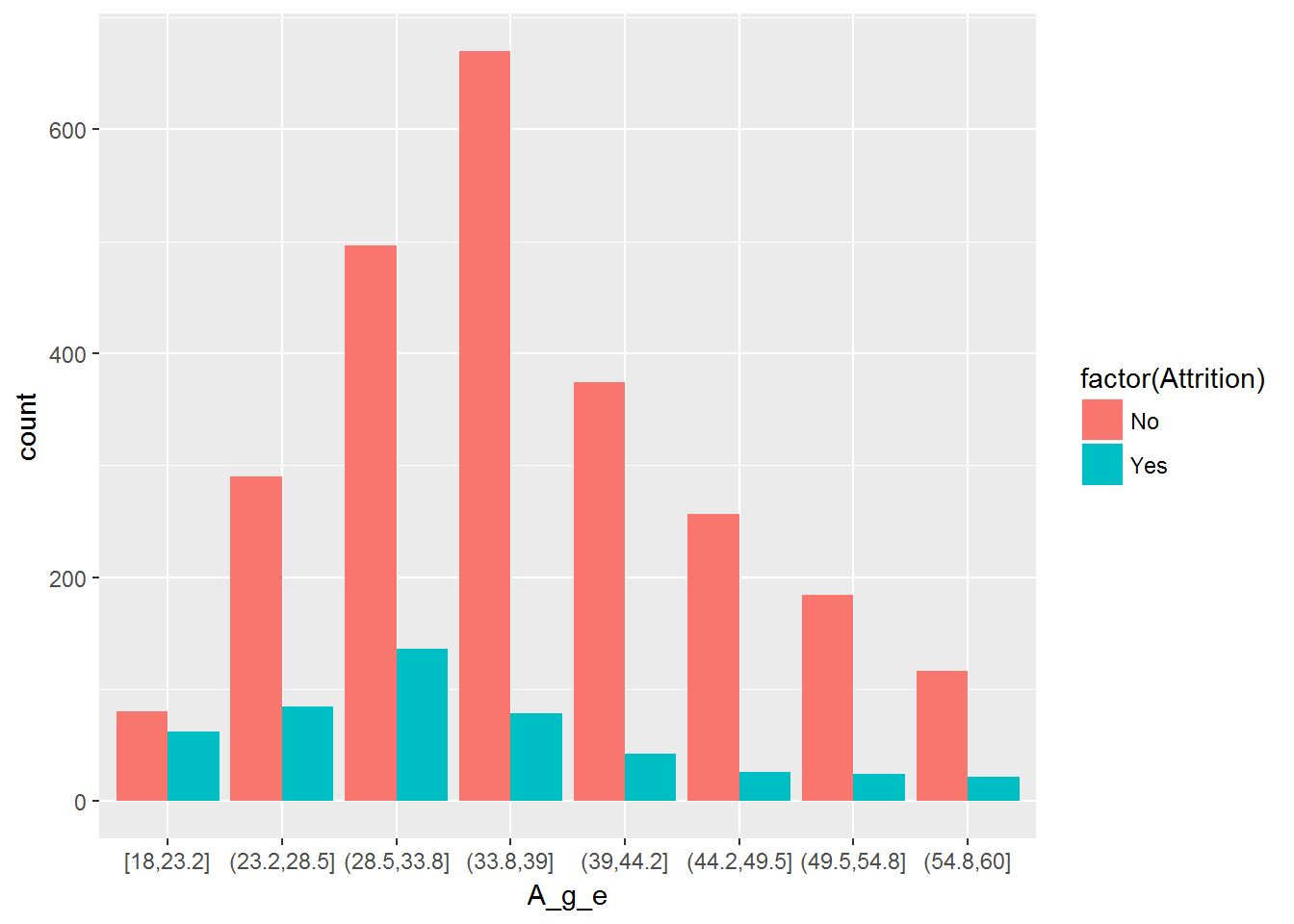
> summary(data$Age)

Min. 1st Qu. Median Mean 3rd Qu. Max.

18.00 30.00 36.00 36.92 43.00 60.00

> A\_g\_e <- cut(data$Age, 8, include.lowest = TRUE)

|  |
| --- |
| > ggplot(data, aes(A\_g\_e, ..count.., fill = factor(Attrition))) + geom\_bar(position="dodge") |
|  |
| |  | | --- | | > | |

****

The age group of 18-23 contributes the maximum to attrition in percentage terms. Post 34, attrition shows a downward trend.

**Modelling:**

**CART Model:**

> data1<-read.csv(file.choose())

> attach(data1)

> View(data1)

> summary(data1)

Age Attrition BusinessTravel DailyRate

Min. :18.00 No :2466 Non-Travel : 300 Min. : 102.0

1st Qu.:30.00 Yes: 474 Travel\_Frequently: 554 1st Qu.: 465.0

Median :36.00 Travel\_Rarely :2086 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 : 126 Min. : 1.000 Min. :1.000

Research & Development:1922 1st Qu.: 2.000 1st Qu.:2.000

Sales : 892 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 : 54 Min. :1 Min. : 1.0 Min. :1.000

Life Sciences :1212 1st Qu.:1 1st Qu.: 735.8 1st Qu.:2.000

Marketing : 318 Median :1 Median :1470.5 Median :3.000

Medical : 928 Mean :1 Mean :1470.5 Mean :2.722

Other : 164 3rd Qu.:1 3rd Qu.:2205.2 3rd Qu.:4.000

Technical Degree: 264 Max. :1 Max. :2940.0 Max. :4.000

Gender HourlyRate JobInvolvement JobLevel

Female:1176 Min. : 30.00 Min. :1.00 Min. :1.000

Male :1764 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.: 84.00 3rd Qu.:3.00 3rd Qu.:3.000

Max. :100.00 Max. :4.00 Max. :5.000

JobRole JobSatisfaction MaritalStatus MonthlyIncome

Sales Executive :652 Min. :1.000 Divorced: 654 Min. : 1009

Research Scientist :584 1st Qu.:2.000 Married :1346 1st Qu.: 2911

Laboratory Technician :518 Median :3.000 Single : 940 Median : 4919

Manufacturing Director :290 Mean :2.729 Mean : 6503

Healthcare Representative:262 3rd Qu.:4.000 3rd Qu.: 8380

Manager :204 Max. :4.000 Max. :19999

(Other) :430

MonthlyRate NumCompaniesWorked Over18 OverTime PercentSalaryHike

Min. : 2094 Min. :0.000 Y:2940 No :2108 Min. :11.00

1st Qu.: 8045 1st Qu.:1.000 Yes: 832 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

> data1$EmployeeCount<-NULL

> data1$EmployeeNumber<-NULL

> data1$Over18<-NULL

> data1$StandardHours<-NULL

> str(data1)

'data.frame': 2940 obs. of 31 variables:

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

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

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

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

$ Department : Factor w/ 3 levels "Human Resources",..: 3 2 2 2 2 2 2 2 2 2 ...

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

$ Education : int 2 1 2 4 1 2 3 1 3 3 ...

$ EducationField : Factor w/ 6 levels "Human Resources",..: 2 2 5 2 4 2 4 2 2 4 ...

$ EnvironmentSatisfaction : int 2 3 4 4 1 4 3 4 4 3 ...

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

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

$ JobInvolvement : int 3 2 2 3 3 3 4 3 2 3 ...

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

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

$ JobSatisfaction : int 4 2 3 3 2 4 1 3 3 3 ...

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

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

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

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

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

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

$ PerformanceRating : int 3 4 3 3 3 3 4 4 4 3 ...

$ RelationshipSatisfaction: int 1 4 2 3 4 3 1 2 2 2 ...

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

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

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

$ WorkLifeBalance : int 1 3 3 3 3 2 2 3 3 2 ...

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

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

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

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

> detach(data1)

> train.data<-read.csv(file.choose())

> attach(train.data)

> str(train.data)

'data.frame': 2940 obs. of 35 variables:

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

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

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

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

$ Department : Factor w/ 3 levels "Human Resources",..: 3 2 2 2 2 2 2 2 2 2 ...

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

$ Education : int 2 1 2 4 1 2 3 1 3 3 ...

$ EducationField : Factor w/ 6 levels "Human Resources",..: 2 2 5 2 4 2 4 2 2 4 ...

$ EmployeeCount : int 1 1 1 1 1 1 1 1 1 1 ...

$ EmployeeNumber : int 1 2 3 4 5 6 7 8 9 10 ...

$ EnvironmentSatisfaction : int 2 3 4 4 1 4 3 4 4 3 ...

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

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

$ JobInvolvement : int 3 2 2 3 3 3 4 3 2 3 ...

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

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

$ JobSatisfaction : int 4 2 3 3 2 4 1 3 3 3 ...

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

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

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

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

$ Over18 : Factor w/ 1 level "Y": 1 1 1 1 1 1 1 1 1 1 ...

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

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

$ PerformanceRating : int 3 4 3 3 3 3 4 4 4 3 ...

$ RelationshipSatisfaction: int 1 4 2 3 4 3 1 2 2 2 ...

$ StandardHours : int 80 80 80 80 80 80 80 80 80 80 ...

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

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

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

$ WorkLifeBalance : int 1 3 3 3 3 2 2 3 3 2 ...

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

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

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

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

> ## loading the library

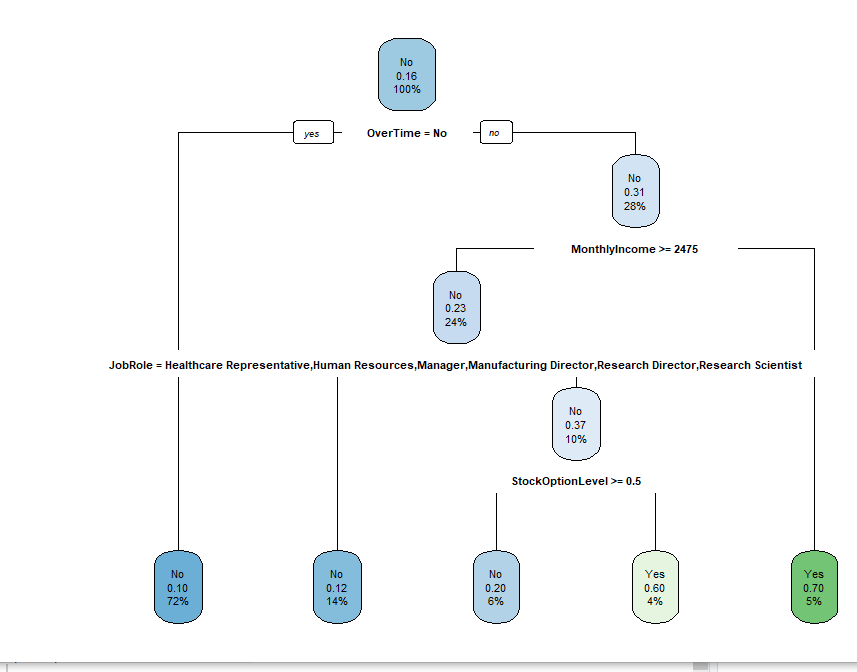
> library(rpart)

> library(rpart.plot)

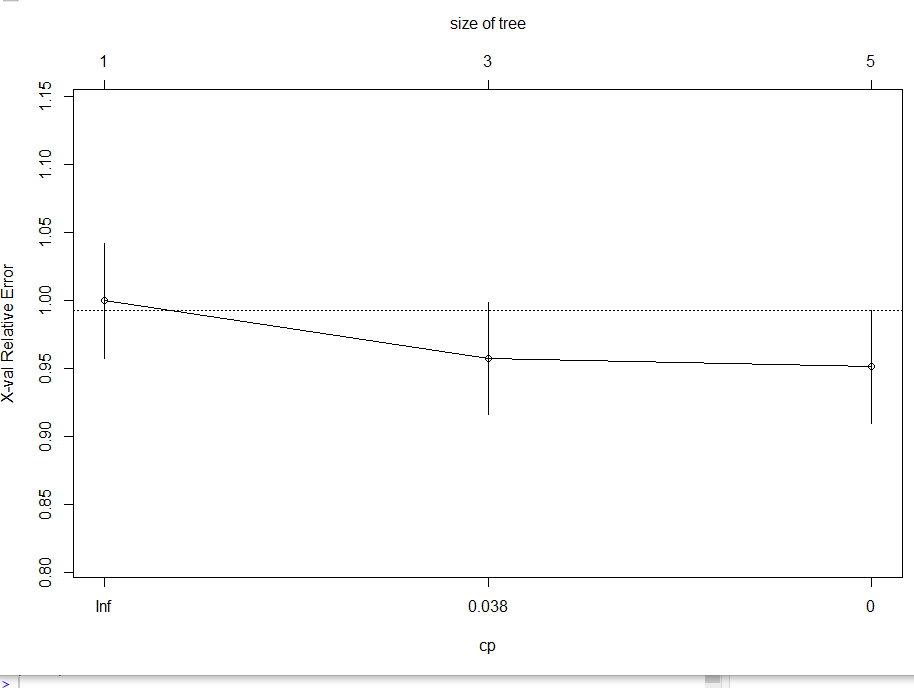
|  |
| --- |
| > ## setting the control parameter inputs for rpart  > r.ctrl = rpart.control(minsplit=235, minbucket = 78, cp = 0, xval = 5)  > ## calling the rpart function to build the tree  > m1 <- rpart(formula = Attrition ~ ., data = train.data[-1], method = "class", control = r.ctrl)  > m1  n= 2940  node), split, n, loss, yval, (yprob)  \* denotes terminal node  1) root 2940 474 No (0.8387755 0.1612245)  2) OverTime=No 2108 220 No (0.8956357 0.1043643) \*  3) OverTime=Yes 832 254 No (0.6947115 0.3052885)  6) MonthlyIncome>=2475 694 158 No (0.7723343 0.2276657)  12) JobRole=Healthcare Representative,Human Resources,Manager,Manufacturing Director,Research Director,Research Scientist 400 50 No (0.8750000 0.1250000) \*  13) JobRole=Laboratory Technician,Sales Executive,Sales Representative 294 108 No (0.6326531 0.3673469)  26) StockOptionLevel>=0.5 170 34 No (0.8000000 0.2000000) \*  27) StockOptionLevel< 0.5 124 50 Yes (0.4032258 0.5967742) \*  7) MonthlyIncome< 2475 138 42 Yes (0.3043478 0.6956522) \* |
|  |
| |  | | --- | | > | |

> #Plot the tree

> rpart.plot(m1,roundint = FALSE)



|  |
| --- |
| > ## to find how the tree performs and derive cp value  > printcp(m1)  Classification tree:  rpart(formula = Attrition ~ ., data = train.data[-1], method = "class",  control = r.ctrl)  Variables actually used in tree construction:  [1] JobRole MonthlyIncome OverTime StockOptionLevel  Root node error: 474/2940 = 0.16122  n= 2940  CP nsplit rel error xerror xstd  1 0.056962 0 1.00000 1.00000 0.042066  2 0.025316 2 0.88608 0.95781 0.041336  3 0.000000 4 0.83544 0.95148 0.041224  > plotcp(m1) |
|  |
| |  | | --- | | > | |



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| > ## Predicting classes and probabilities  > class <- predict(m1, train.data[-1], type="class")  > prob <- predict(m1, train.data[-1])  > class  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  Yes No Yes No No No No No No No No Yes No No Yes No  17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32  No No No No No No No No No No No No No No No No  33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48  No No Yes No Yes No No No No No No No No No No No  49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64  Yes No Yes Yes No No No No No No No No No No No No  65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80  No No No No No No No No No No No No No No No No  81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96  No No No No No No No No No No No Yes No No No No  97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112  No No No No Yes No Yes No No No No Yes No No No Yes  113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128  No No No No No No No No No No No No Yes No No Yes  129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144  No No No No No No No No No No No No No No No No  145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160  No No No No No No No No Yes No No No No No No No  161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176  No No No No No No No No No No No No No No No No  177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192  No No No No No No Yes No No No No No No No No No  193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208  No No No No No No No No No No No No No No Yes No  209 003210 211 212 213 214 215 216 217 218 219 220 221 222 223 224  No No No No No No No No No No No No No No No No  225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240  No No No No No Yes No No No No Yes No No No No Yes  241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256  No No No No No No No No No No No No No No No No  257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272  No No No No No No No No No No No No No No No No  273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288  No No No No No No No No No No No No No No No No  289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304  Yes No No No No Yes Yes No No No No No No No No No  305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320  No No No No No No No No No No No No No No No Yes  321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336  Yes No No No No No No Yes No No No No No No No No  337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352  Yes No No No No No No No No No No No No No No No  353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368  No No No No No Yes No No No Yes No Yes No No No No  369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384  No No No No No No No No No No Yes No No No No No  385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400  No Yes No No No No No No No No No No No Yes No No  401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416  No No No No No No No No No No No No No No Yes No  417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432  Yes No No No No No No No No No No No No No No No  433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448  No No No No No No No No No No No No No No No No  449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464  No No Yes No No No No No No Yes No No No No No Yes  465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480  No No No No No No No No No No No No No No No No  481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496  No No No No No No No No No No No No No No No No  497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512  No No No No No No No No No No No No No No No No  513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528  No Yes No No No No No No No No No No No No No No  529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544  No No No No No No No No No No No No Yes No No No  545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560  No No No No No No No No No Yes No No No No No No  561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576  No No No No No No Yes No No No No No No No No No  577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592  No No No No No No No No No Yes No No No Yes No No  593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608  No No No No No No No No No No No No No No No No  609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624  No No No No Yes No Yes No No No No No No No No No  625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640  No No No No No No No No No No Yes No Yes No No No  641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656  No No No No Yes No No No No No No No Yes No No Yes  657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672  Yes No No No Yes No No No No Yes No No No Yes No No  673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688  No No No No No No No No No No No Yes No No Yes No  689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704  [ reached getOption("max.print") -- omitted 1940 entries ]  Levels: No Yes  > prob  No Yes  1 0.4032258 0.5967742  2 0.8956357 0.1043643  3 0.3043478 0.6956522  4 0.8750000 0.1250000  5 0.8956357 0.1043643  6 0.8956357 0.1043643  7 0.8000000 0.2000000  8 0.8956357 0.1043643  9 0.8956357 0.1043643  10 0.8956357 0.1043643  11 0.8956357 0.1043643  12 0.4032258 0.5967742  13 0.8956357 0.1043643  14 0.8956357 0.1043643  15 0.3043478 0.6956522  16 0.8956357 0.1043643  17 0.8750000 0.1250000  18 0.8000000 0.2000000  19 0.8956357 0.1043643  20 0.8750000 0.1250000  21 0.8956357 0.1043643  22 0.8956357 0.1043643  23 0.8956357 0.1043643  24 0.8956357 0.1043643  25 0.8956357 0.1043643  26 0.8956357 0.1043643  27 0.8750000 0.1250000  28 0.8956357 0.1043643  29 0.8956357 0.1043643  30 0.8956357 0.1043643  31 0.8956357 0.1043643  32 0.8750000 0.1250000  33 0.8956357 0.1043643  34 0.8956357 0.1043643  35 0.3043478 0.6956522  36 0.8956357 0.1043643  37 0.4032258 0.5967742  38 0.8956357 0.1043643  39 0.8750000 0.1250000  40 0.8956357 0.1043643  41 0.8956357 0.1043643  42 0.8956357 0.1043643  43 0.8956357 0.1043643  44 0.8956357 0.1043643  45 0.8956357 0.1043643  46 0.8956357 0.1043643  47 0.8956357 0.1043643  48 0.8956357 0.1043643  49 0.4032258 0.5967742  50 0.8956357 0.1043643  51 0.4032258 0.5967742  52 0.4032258 0.5967742  53 0.8000000 0.2000000  54 0.8750000 0.1250000  55 0.8000000 0.2000000  56 0.8750000 0.1250000  57 0.8956357 0.1043643  58 0.8000000 0.2000000  59 0.8956357 0.1043643  60 0.8956357 0.1043643  61 0.8750000 0.1250000  62 0.8956357 0.1043643  63 0.8750000 0.1250000  64 0.8956357 0.1043643  65 0.8956357 0.1043643  66 0.8750000 0.1250000  67 0.8956357 0.1043643  68 0.8956357 0.1043643  69 0.8956357 0.1043643  70 0.8750000 0.1250000  71 0.8956357 0.1043643  72 0.8956357 0.1043643  73 0.8956357 0.1043643  74 0.8956357 0.1043643  75 0.8956357 0.1043643  76 0.8956357 0.1043643  77 0.8956357 0.1043643  78 0.8750000 0.1250000  79 0.8956357 0.1043643  80 0.8750000 0.1250000  81 0.8000000 0.2000000  82 0.8956357 0.1043643  83 0.8956357 0.1043643  84 0.8750000 0.1250000  85 0.8956357 0.1043643  86 0.8956357 0.1043643  87 0.8956357 0.1043643  88 0.8956357 0.1043643  89 0.8956357 0.1043643  90 0.8956357 0.1043643  91 0.8956357 0.1043643  92 0.4032258 0.5967742  93 0.8000000 0.2000000  94 0.8750000 0.1250000  95 0.8956357 0.1043643  96 0.8956357 0.1043643  97 0.8956357 0.1043643  98 0.8956357 0.1043643  99 0.8956357 0.1043643  100 0.8956357 0.1043643  101 0.3043478 0.6956522  102 0.8956357 0.1043643  103 0.4032258 0.5967742  104 0.8956357 0.1043643  105 0.8956357 0.1043643  106 0.8956357 0.1043643  107 0.8750000 0.1250000  108 0.4032258 0.5967742  109 0.8956357 0.1043643  110 0.8956357 0.1043643  111 0.8956357 0.1043643  112 0.4032258 0.5967742  113 0.8750000 0.1250000  114 0.8956357 0.1043643  115 0.8956357 0.1043643  116 0.8956357 0.1043643  117 0.8956357 0.1043643  118 0.8956357 0.1043643  119 0.8956357 0.1043643  120 0.8750000 0.1250000  121 0.8956357 0.1043643  122 0.8956357 0.1043643  123 0.8750000 0.1250000  124 0.8956357 0.1043643  125 0.4032258 0.5967742  126 0.8956357 0.1043643  127 0.8956357 0.1043643  128 0.3043478 0.6956522  129 0.8956357 0.1043643  130 0.8956357 0.1043643  131 0.8956357 0.1043643  132 0.8956357 0.1043643  133 0.8000000 0.2000000  134 0.8000000 0.2000000  135 0.8956357 0.1043643  136 0.8956357 0.1043643  137 0.8956357 0.1043643  138 0.8956357 0.1043643  139 0.8956357 0.1043643  140 0.8750000 0.1250000  141 0.8956357 0.1043643  142 0.8956357 0.1043643  143 0.8750000 0.1250000  144 0.8956357 0.1043643  145 0.8956357 0.1043643  146 0.8956357 0.1043643  147 0.8956357 0.1043643  148 0.8956357 0.1043643  149 0.8956357 0.1043643  150 0.8956357 0.1043643  151 0.8956357 0.1043643  152 0.8956357 0.1043643  153 0.3043478 0.6956522  154 0.8956357 0.1043643  155 0.8956357 0.1043643  156 0.8956357 0.1043643  157 0.8956357 0.1043643  158 0.8750000 0.1250000  159 0.8956357 0.1043643  160 0.8956357 0.1043643  161 0.8956357 0.1043643  162 0.8956357 0.1043643  163 0.8956357 0.1043643  164 0.8750000 0.1250000  165 0.8750000 0.1250000  166 0.8956357 0.1043643  167 0.8956357 0.1043643  168 0.8956357 0.1043643  169 0.8956357 0.1043643  170 0.8956357 0.1043643  171 0.8750000 0.1250000  172 0.8956357 0.1043643  173 0.8956357 0.1043643  174 0.8956357 0.1043643  175 0.8000000 0.2000000  176 0.8750000 0.1250000  177 0.8956357 0.1043643  178 0.8956357 0.1043643  179 0.8956357 0.1043643  180 0.8956357 0.1043643  181 0.8750000 0.1250000  182 0.8956357 0.1043643  183 0.4032258 0.5967742  184 0.8956357 0.1043643  185 0.8956357 0.1043643  186 0.8956357 0.1043643  187 0.8956357 0.1043643  188 0.8956357 0.1043643  189 0.8956357 0.1043643  190 0.8956357 0.1043643  191 0.8956357 0.1043643  192 0.8956357 0.1043643  193 0.8750000 0.1250000  194 0.8956357 0.1043643  195 0.8956357 0.1043643  196 0.8750000 0.1250000  197 0.8956357 0.1043643  198 0.8956357 0.1043643  199 0.8956357 0.1043643  200 0.8956357 0.1043643  201 0.8956357 0.1043643  202 0.8750000 0.1250000  203 0.8750000 0.1250000  204 0.8000000 0.2000000  205 0.8750000 0.1250000  206 0.8956357 0.1043643  207 0.3043478 0.6956522  208 0.8956357 0.1043643  209 0.8956357 0.1043643  210 0.8956357 0.1043643  211 0.8956357 0.1043643  212 0.8956357 0.1043643  213 0.8956357 0.1043643  214 0.8956357 0.1043643  215 0.8750000 0.1250000  216 0.8750000 0.1250000  217 0.8956357 0.1043643  218 0.8956357 0.1043643  219 0.8956357 0.1043643  220 0.8956357 0.1043643  221 0.8956357 0.1043643  222 0.8956357 0.1043643  223 0.8750000 0.1250000  224 0.8956357 0.1043643  225 0.8956357 0.1043643  226 0.8956357 0.1043643  227 0.8956357 0.1043643  228 0.8956357 0.1043643  229 0.8956357 0.1043643  230 0.3043478 0.6956522  231 0.8956357 0.1043643  232 0.8956357 0.1043643  233 0.8956357 0.1043643  234 0.8956357 0.1043643  235 0.3043478 0.6956522  236 0.8750000 0.1250000  237 0.8956357 0.1043643  238 0.8750000 0.1250000  239 0.8956357 0.1043643  240 0.4032258 0.5967742  241 0.8956357 0.1043643  242 0.8956357 0.1043643  243 0.8956357 0.1043643  244 0.8956357 0.1043643  245 0.8956357 0.1043643  246 0.8956357 0.1043643  247 0.8956357 0.1043643  248 0.8956357 0.1043643  249 0.8956357 0.1043643  250 0.8956357 0.1043643   |  | | --- | | > library(caret)  > confusionMatrix(train.data$Attrition,train.data$predicted.Dtree,mode = "everything")  Confusion Matrix and Statistics  Reference  Prediction No Yes  No 2374 92  Yes 304 170    Accuracy : 0.8653  95% CI : (0.8524, 0.8774)  No Information Rate : 0.9109  P-Value [Acc > NIR] : 1    Kappa : 0.3922    Mcnemar's Test P-Value : <2e-16    Sensitivity : 0.8865  Specificity : 0.6489  Pos Pred Value : 0.9627  Neg Pred Value : 0.3586  Precision : 0.9627  Recall : 0.8865  F1 : 0.9230  Prevalence : 0.9109  Detection Rate : 0.8075  Detection Prevalence : 0.8388  Balanced Accuracy : 0.7677    'Positive' Class : No | |  | | **Neural Network:** Let us create a Neural Network model with the same data: We used nnet function to generate neural network model with train dataset. It went for 750 iteration and converged. Accuracy of model is 0.879 which is better than the CART model.   |  | | --- | | > nn1 <- nnet(formula = Attrition ~ .,  + data = traindf,  + size=14,rang=0.01,Hess=FALSE,decay=0.0,maxit=2000)  # weights: 617  initial value 1405.444657  iter 10 value 876.294050  iter 20 value 857.226239  iter 30 value 828.766875  iter 40 value 824.110105  iter 50 value 816.462086  iter 60 value 804.511935  iter 70 value 795.648138  iter 80 value 775.629981  iter 90 value 752.164205  iter 100 value 712.788231  iter 110 value 693.957048  iter 120 value 659.587528  iter 130 value 651.623347  iter 140 value 647.115127  iter 150 value 642.103118  iter 160 value 641.715501  iter 170 value 641.055390  iter 180 value 640.299459  iter 190 value 640.263580  iter 200 value 640.251228  iter 210 value 640.232920  iter 220 value 639.620394  iter 230 value 639.401776  iter 240 value 639.239048  iter 250 value 637.547090  iter 260 value 637.385742  iter 270 value 637.223078  iter 280 value 637.138109  iter 290 value 637.079440  iter 300 value 637.047544  iter 310 value 637.026539  iter 320 value 637.020424  iter 330 value 637.017200  iter 340 value 637.013818  iter 350 value 637.011333  iter 360 value 637.009432  iter 370 value 635.740894  iter 380 value 634.331347  iter 390 value 632.859524  iter 400 value 628.525331  iter 410 value 628.401452  iter 420 value 628.388203  iter 430 value 628.378922  iter 440 value 628.375157  iter 450 value 628.372634  iter 460 value 628.370085  iter 470 value 627.707887  iter 480 value 627.497681  iter 490 value 627.466032  iter 500 value 627.458367  iter 510 value 627.454907  iter 520 value 627.453517  iter 530 value 627.450986  iter 540 value 627.448332  iter 550 value 627.445106  iter 560 value 627.441327  iter 570 value 627.232260  iter 580 value 626.282106  iter 590 value 625.617244  iter 600 value 625.575283  iter 610 value 625.559613  iter 620 value 625.547856  iter 630 value 625.542285  iter 640 value 625.538489  iter 650 value 625.536312  iter 660 value 625.534818  iter 670 value 625.533609  iter 680 value 625.532367  iter 690 value 625.530958  iter 700 value 625.529155  iter 710 value 625.527981  iter 720 value 625.527237  iter 730 value 625.525742  iter 740 value 625.517729  iter 750 value 625.515158  final value 625.513656  converged  >  > testnn = predict(nn1,testdf,type=("class"))  > table(testnn)  testnn  No Yes  754 128  > t3= table(testdf$Attrition,testnn)  > t3  testnn  No Yes  No 693 46  Yes 61 82  Accuracy of the Neural Network model = (693+82)/882 = 0.879 | |  | | |  | | --- | | Summary; Overall the Neural Network was better than the CART model in terms of accuracy.  Since the Attrition was set No in 83% of records, there is an imbalance in the target variable.  This is can be offset by oversampling for Attrition to Yes and remove the imbalance. | |   Accuracy of the Neural Network model = (693+82)/882 = 0.879  Accuracy of CART model= (729+27)/882 = 0.857   |  | | --- | |  | | |
|  |
| |  | | --- | |  | |

**Validate Cart Model:**

Confusion Matrix:

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 734 | 24 |
| 1 | 74 | 50 |

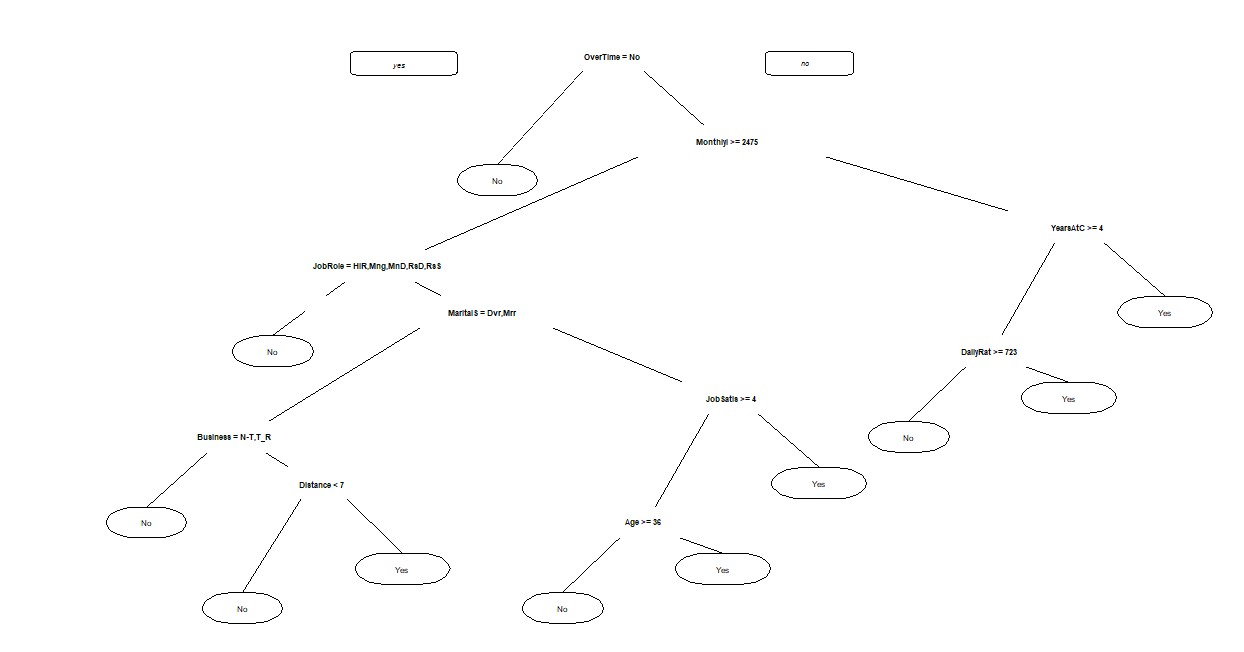
|  |  |
| --- | --- |
| **Confusion Matrix-CART** | |
| Accuracy=(TP+TN)/Total | 0.8653 |

**Pruning:**

> bestcp <- model$cptable[which.min(model$cptable[,"xerror"]),"CP"]

> prunedModel <-prune(model, cp= bestcp)

> prp(prunedModel)



**Classification tree:**

rpart(formula = Attrition ~ ., data = trainD, method = "class")

Variables actually used in tree construction:

[1] Age BusinessTravel DailyRate DistanceFromHome

[5] EnvironmentSatisfaction JobRole JobSatisfaction MaritalStatus

[9] MonthlyIncome MonthlyRate OverTime TotalWorkingYears

[13] YearsAtCompany

Root node error: 350/2058 = 0.17007

n= 2058

CP nsplit rel error xerror xstd

1 0.057143 0 1.00000 1.00000 0.048695

2 0.034286 2 0.88571 0.96000 0.047907

3 0.018571 4 0.81714 0.90000 0.046667

4 0.015714 6 0.78000 0.85143 0.045611

5 0.014286 8 0.74857 0.84286 0.045420

6 0.013333 10 0.72000 0.83429 0.045227

7 0.010000 14 0.66286 0.83429 0.045227

> plotcp(model)

> rpart.plot(model)

> prp(model)

> plotcp(model)

> bestcp <- model$cptable[which.min(model$cptable[,"xerror"]),"CP"]

> prunedModel <-prune(model, cp= bestcp)

> prp(prunedModel)

> prediction\_pm <- predict(prunedModel, newdata=testD, type="class")

> table(testD$Attrition, prediction\_pm)

prediction\_pm

No Yes

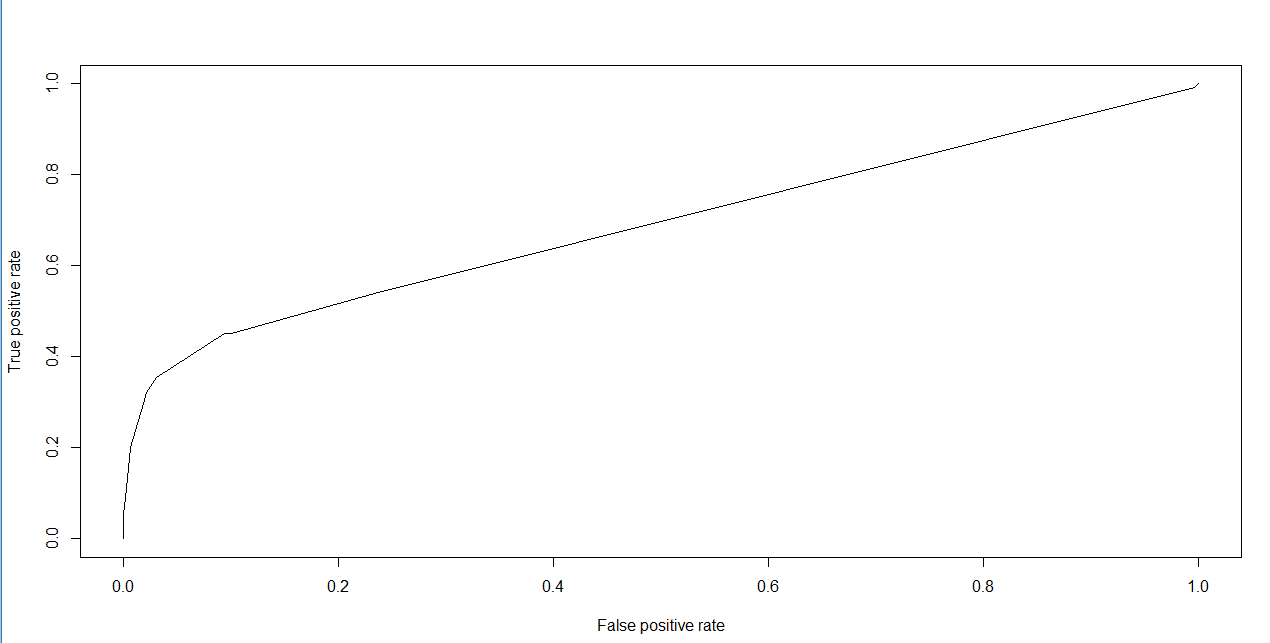
No 742 16

Yes 84 40

> (711+54)/(nrow(testD))

**[1] 0.8673469**

> install.packages("ROCR")



**Post pruning:**

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 742 | 16 |
| 1 | 84 | 40 |

|  |  |
| --- | --- |
| **Confusion Matrix-CART** | |
| Accuracy=(TP+TN)/Total | **0.8673469** |

CONCLUSION:

So accuracy=86.73% is slightly improved post pruning and results are almost same.

**Neural Network:** Let us create a Neural Network model with the same data: We used nnet function to generate neural network model with train dataset. It went for 750 iteration and converged. Accuracy of model is 0.879 which is better than the CART model.

|  |
| --- |
| > nn1 <- nnet(formula = Attrition ~ .,  + data = traindf,  + size=14,rang=0.01,Hess=FALSE,decay=0.0,maxit=2000)  # weights: 617  initial value 1405.444657  iter 10 value 876.294050  iter 20 value 857.226239  iter 30 value 828.766875  iter 40 value 824.110105  iter 50 value 816.462086  iter 60 value 804.511935  iter 70 value 795.648138  iter 80 value 775.629981  iter 90 value 752.164205  iter 100 value 712.788231  iter 110 value 693.957048  iter 120 value 659.587528  iter 130 value 651.623347  iter 140 value 647.115127  iter 150 value 642.103118  iter 160 value 641.715501  iter 170 value 641.055390  iter 180 value 640.299459  iter 190 value 640.263580  iter 200 value 640.251228  iter 210 value 640.232920  iter 220 value 639.620394  iter 230 value 639.401776  iter 240 value 639.239048  iter 250 value 637.547090  iter 260 value 637.385742  iter 270 value 637.223078  iter 280 value 637.138109  iter 290 value 637.079440  iter 300 value 637.047544  iter 310 value 637.026539  iter 320 value 637.020424  iter 330 value 637.017200  iter 340 value 637.013818  iter 350 value 637.011333  iter 360 value 637.009432  iter 370 value 635.740894  iter 380 value 634.331347  iter 390 value 632.859524  iter 400 value 628.525331  iter 410 value 628.401452  iter 420 value 628.388203  iter 430 value 628.378922  iter 440 value 628.375157  iter 450 value 628.372634  iter 460 value 628.370085  iter 470 value 627.707887  iter 480 value 627.497681  iter 490 value 627.466032  iter 500 value 627.458367  iter 510 value 627.454907  iter 520 value 627.453517  iter 530 value 627.450986  iter 540 value 627.448332  iter 550 value 627.445106  iter 560 value 627.441327  iter 570 value 627.232260  iter 580 value 626.282106  iter 590 value 625.617244  iter 600 value 625.575283  iter 610 value 625.559613  iter 620 value 625.547856  iter 630 value 625.542285  iter 640 value 625.538489  iter 650 value 625.536312  iter 660 value 625.534818  iter 670 value 625.533609  iter 680 value 625.532367  iter 690 value 625.530958  iter 700 value 625.529155  iter 710 value 625.527981  iter 720 value 625.527237  iter 730 value 625.525742  iter 740 value 625.517729  iter 750 value 625.515158  final value 625.513656  converged  >  > testnn = predict(nn1,testdf,type=("class"))  > table(testnn)  testnn  No Yes  754 128  > t3= table(testdf$Attrition,testnn)  > t3  testnn  No Yes  No 693 46  Yes 61 82  Accuracy of the Neural Network model = (693+82)/882 = 0.879 |

Summary: Overall Neural Networks and CART model have closer accuracy. There is not big change due to Neural Networks.