HR Attrition using CART, RF and ANN models

PVS

Wednesday, January 25, 2017

## Problem Statement

Please find below the Assignment on HR employee attrition. The csv file is attached herewith:HR\_Employee\_Attrition\_Data-1.csvView in a new window

## Solution

## Steps:

1. Import the **HR\_Employee\_Attrition\_Data.csv file** in R
2. See the structure of the file using *str()* function in R
3. Perform EDA of the data using *summary()* function
4. Convert continuous variables into Categorical Variables having 5 buckets. You will have to write if-else logic in R. (Note: Attrition column in the dataset is your Target Column. EmployeeNumber is the Identifier column.)
5. Compute CHI-SQ statistics for all the independent categorical variable vs Attrition variable

Sr. No | Variable | Chi-SQ | Degree of Freedom | P-Value

1. Based on CHI-SQ statistics find the important variables
2. Build Model on HR Employee Attrition Data using "CART", "Random Forest" and "Neural Network" w.r.t. your output in step 6
3. Try to interpret your Classification Tree output
4. Compare model performance
5. While building the model they may see the model performance measures are not holding on the Hold-Out Sample

Tip 1: Reduce the Cross Validation for CART Tip 2: Increase the Error Threshold Tip 3: Take Dev & Hold-Out sample as 70% of the Population. Some records in Hold-out and Dev will overlap

## Objective

The objective of this study is to reduce HR attrition. We want to identify the sub-group of employees who are likely to leave the organization.

### Load required libraries

library(partykit)

## Loading required package: grid

library(caret)

## Warning: package 'caret' was built under R version 3.2.5

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.2.5

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.2.5

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(rpart)  
library(rpart.plot)  
library(rattle)

## Rattle: A free graphical interface for data mining with R.  
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(RColorBrewer)  
library(plyr)

## Warning: package 'plyr' was built under R version 3.2.5

library(ROCR)

## Warning: package 'ROCR' was built under R version 3.2.5

## Loading required package: gplots

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

library(ineq)

## Warning: package 'ineq' was built under R version 3.2.5

if(!require(dplyr)){  
 install.packages("dplyr")  
}

## Loading required package: dplyr

## Warning: package 'dplyr' was built under R version 3.2.5

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

## The following object is masked from 'package:randomForest':  
##   
## combine

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(dplyr)

### 1) Import the HR\_Employee\_Attrition\_Data.csv file in R

### 2) See the structure of the file using *str()* function in R

Let us view the internal structure of the R object, hrdata

str(hrdata)

## '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 ...

# Define a function to find the type of variable in the data frame, hrData  
  
findClass <- function (x) {  
 for (i in 1:35)  
 {  
 l1 <- names(x[i])  
 l2 <- class(x[[i]])  
 cat("\n ",l1,": ",l2)  
 }   
}  
  
#  
cat("\n Variables according to the class \n")

##   
## Variables according to the class

findClass(hrdata) ## Get the list of type of variables

##   
## Age : integer  
## Attrition : factor  
## BusinessTravel : factor  
## DailyRate : integer  
## Department : factor  
## DistanceFromHome : integer  
## Education : integer  
## EducationField : factor  
## EmployeeCount : integer  
## EmployeeNumber : integer  
## EnvironmentSatisfaction : integer  
## Gender : factor  
## HourlyRate : integer  
## JobInvolvement : integer  
## JobLevel : integer  
## JobRole : factor  
## JobSatisfaction : integer  
## MaritalStatus : factor  
## MonthlyIncome : integer  
## MonthlyRate : integer  
## NumCompaniesWorked : integer  
## Over18 : factor  
## OverTime : factor  
## PercentSalaryHike : integer  
## PerformanceRating : integer  
## RelationshipSatisfaction : integer  
## StandardHours : integer  
## StockOptionLevel : integer  
## TotalWorkingYears : integer  
## TrainingTimesLastYear : integer  
## WorkLifeBalance : integer  
## YearsAtCompany : integer  
## YearsInCurrentRole : integer  
## YearsSinceLastPromotion : integer  
## YearsWithCurrManager : integer

#  
cat("\n Variables with number of missing values \n")

##   
## Variables with number of missing values

sapply(hrdata, function(x) sum(is.na(x))) # To report missing values

## Age Attrition BusinessTravel   
## 0 0 0   
## DailyRate Department DistanceFromHome   
## 0 0 0   
## Education EducationField EmployeeCount   
## 0 0 0   
## EmployeeNumber EnvironmentSatisfaction Gender   
## 0 0 0   
## HourlyRate JobInvolvement JobLevel   
## 0 0 0   
## JobRole JobSatisfaction MaritalStatus   
## 0 0 0   
## MonthlyIncome MonthlyRate NumCompaniesWorked   
## 0 0 0   
## Over18 OverTime PercentSalaryHike   
## 0 0 0   
## PerformanceRating RelationshipSatisfaction StandardHours   
## 0 0 0   
## StockOptionLevel TotalWorkingYears TrainingTimesLastYear   
## 0 0 0   
## WorkLifeBalance YearsAtCompany YearsInCurrentRole   
## 0 0 0   
## YearsSinceLastPromotion YearsWithCurrManager   
## 0 0

**Observation**

There are 35 variables and 2940 observations with no missing values.

Attrition is a target variable. Rest are predictor variables.

**A) Numerical variables:**

1. Age
2. DailyRate
3. HourlyRate
4. MonthlyIncome
5. MonthlyRate
6. DistanceFromHome
7. EmployeeNumber (ID variable)
8. NumCompaniesWorked
9. PresentSalaryHike
10. StandardHours
11. TotalWorkingYears
12. TrainingTimesLastYear
13. YearsAtCompany
14. YearsinCurrentRole
15. YearsSinceLastPromotion
16. YearsWithCurrManager

**B) Integer variables -categorical**

1. Education
2. EmployeeCount
3. EnvironmentSatisfaction
4. JobInvolvement
5. JobLevel
6. JobSatisfaction
7. PerformanceRating
8. RelationshipSatisfaction
9. StockOptionLevel
10. WorkLifeBalance

**C) Factor variables**

1. Attrition
2. BusinessTravel
3. Department
4. Gender
5. EducationField
6. JobRole
7. MaritalStatus
8. Over18
9. OverTime

### 3) Perform EDA of the data using *summary()* function

## 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   
## Human Resources : 54 Min. :1 Min. : 1.0   
## Life Sciences :1212 1st Qu.:1 1st Qu.: 735.8   
## Marketing : 318 Median :1 Median :1470.5   
## Medical : 928 Mean :1 Mean :1470.5   
## Other : 164 3rd Qu.:1 3rd Qu.:2205.2   
## Technical Degree: 264 Max. :1 Max. :2940.0   
##   
## EnvironmentSatisfaction Gender HourlyRate JobInvolvement  
## Min. :1.000 Female:1176 Min. : 30.00 Min. :1.00   
## 1st Qu.:2.000 Male :1764 1st Qu.: 48.00 1st Qu.:2.00   
## Median :3.000 Median : 66.00 Median :3.00   
## Mean :2.722 Mean : 65.89 Mean :2.73   
## 3rd Qu.:4.000 3rd Qu.: 84.00 3rd Qu.:3.00   
## Max. :4.000 Max. :100.00 Max. :4.00   
##   
## JobLevel JobRole JobSatisfaction  
## Min. :1.000 Sales Executive :652 Min. :1.000   
## 1st Qu.:1.000 Research Scientist :584 1st Qu.:2.000   
## Median :2.000 Laboratory Technician :518 Median :3.000   
## Mean :2.064 Manufacturing Director :290 Mean :2.729   
## 3rd Qu.:3.000 Healthcare Representative:262 3rd Qu.:4.000   
## Max. :5.000 Manager :204 Max. :4.000   
## (Other) :430   
## MaritalStatus MonthlyIncome MonthlyRate NumCompaniesWorked  
## Divorced: 654 Min. : 1009 Min. : 2094 Min. :0.000   
## Married :1346 1st Qu.: 2911 1st Qu.: 8045 1st Qu.:1.000   
## Single : 940 Median : 4919 Median :14236 Median :2.000   
## Mean : 6503 Mean :14313 Mean :2.693   
## 3rd Qu.: 8380 3rd Qu.:20462 3rd Qu.:4.000   
## Max. :19999 Max. :26999 Max. :9.000   
##   
## Over18 OverTime PercentSalaryHike PerformanceRating  
## Y:2940 No :2108 Min. :11.00 Min. :3.000   
## Yes: 832 1st Qu.:12.00 1st Qu.:3.000   
## Median :14.00 Median :3.000   
## Mean :15.21 Mean :3.154   
## 3rd Qu.:18.00 3rd Qu.:3.000   
## Max. :25.00 Max. :4.000   
##   
## RelationshipSatisfaction StandardHours StockOptionLevel TotalWorkingYears  
## Min. :1.000 Min. :80 Min. :0.0000 Min. : 0.00   
## 1st Qu.:2.000 1st Qu.:80 1st Qu.:0.0000 1st Qu.: 6.00   
## Median :3.000 Median :80 Median :1.0000 Median :10.00   
## Mean :2.712 Mean :80 Mean :0.7939 Mean :11.28   
## 3rd Qu.:4.000 3rd Qu.:80 3rd Qu.:1.0000 3rd Qu.:15.00   
## Max. :4.000 Max. :80 Max. :3.0000 Max. :40.00   
##   
## TrainingTimesLastYear WorkLifeBalance YearsAtCompany YearsInCurrentRole  
## Min. :0.000 Min. :1.000 Min. : 0.000 Min. : 0.000   
## 1st Qu.:2.000 1st Qu.:2.000 1st Qu.: 3.000 1st Qu.: 2.000   
## Median :3.000 Median :3.000 Median : 5.000 Median : 3.000   
## Mean :2.799 Mean :2.761 Mean : 7.008 Mean : 4.229   
## 3rd Qu.:3.000 3rd Qu.:3.000 3rd Qu.: 9.000 3rd Qu.: 7.000   
## Max. :6.000 Max. :4.000 Max. :40.000 Max. :18.000   
##   
## YearsSinceLastPromotion YearsWithCurrManager  
## Min. : 0.000 Min. : 0.000   
## 1st Qu.: 0.000 1st Qu.: 2.000   
## Median : 1.000 Median : 3.000   
## Mean : 2.188 Mean : 4.123   
## 3rd Qu.: 3.000 3rd Qu.: 7.000   
## Max. :15.000 Max. :17.000   
##

**Observation**

* By looking at the output from the **summary** function, we infer the following:
* Minimum, Maximum, Mean, First Quartile, Median and Third Quartile values of the following variables are same indicating the contant 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!

Obviously we do not need the above variables.

## '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 ...

### Let us form the data frames for numeric columns to study any relationship exists among them.

## Age DailyRate HourlyRate MonthlyIncome  
## Age 1.00000000 0.010660943 0.02428654 0.497854567  
## DailyRate 0.01066094 1.000000000 0.02338142 0.007707059  
## HourlyRate 0.02428654 0.023381422 1.00000000 -0.015794304  
## MonthlyIncome 0.49785457 0.007707059 -0.01579430 1.000000000  
## MonthlyRate 0.02805117 -0.032181602 -0.01529675 0.034813626  
## MonthlyRate  
## Age 0.02805117  
## DailyRate -0.03218160  
## HourlyRate -0.01529675  
## MonthlyIncome 0.03481363  
## MonthlyRate 1.00000000

## DistanceFromHome NumCompaniesWorked  
## DistanceFromHome 1.000000000 -0.02925080  
## NumCompaniesWorked -0.029250804 1.00000000  
## PercentSalaryHike 0.040235377 -0.01023831  
## TotalWorkingYears 0.004628426 0.23763859  
## TrainingTimesLastYear -0.036942234 -0.06605407  
## PercentSalaryHike TotalWorkingYears  
## DistanceFromHome 0.040235377 0.004628426  
## NumCompaniesWorked -0.010238309 0.237638590  
## PercentSalaryHike 1.000000000 -0.020608488  
## TotalWorkingYears -0.020608488 1.000000000  
## TrainingTimesLastYear -0.005221012 -0.035661571  
## TrainingTimesLastYear  
## DistanceFromHome -0.036942234  
## NumCompaniesWorked -0.066054072  
## PercentSalaryHike -0.005221012  
## TotalWorkingYears -0.035661571  
## TrainingTimesLastYear 1.000000000

## YearsAtCompany YearsInCurrentRole  
## YearsAtCompany 1.0000000 0.7587537  
## YearsInCurrentRole 0.7587537 1.0000000  
## YearsSinceLastPromotion 0.6184089 0.5480562  
## YearsWithCurrManager 0.7692124 0.7143648  
## YearsSinceLastPromotion YearsWithCurrManager  
## YearsAtCompany 0.6184089 0.7692124  
## YearsInCurrentRole 0.5480562 0.7143648  
## YearsSinceLastPromotion 1.0000000 0.5102236  
## YearsWithCurrManager 0.5102236 1.0000000

**Observation**

There is linear relationship among the continuous numeric variables as explained below:

**List of variables**: 1) Age,DailyRate, HourlyRate,MonthlyIncome,MonthlyRate - MonthlyIncome and Age with Correlation coefficient = 0.49 - Among other variables, maximum correlation coefficient is 0.007707 indicating that these variables have no linear relationship among them.

1. DistanceFromHome,NumCompaniesWorked,PercentSalaryHike,TotalWorkingYears,TrainingTimesLastYear

* The maximum correlation coefficient among the variables in this list is NumCompaniesWorked and TotalWorkingYears is 0.23763859

No Correlation exists among the variables in this list.

1. YearsAtCompany,YearsInCurrentRole,YearsSinceLastPromotion,YearsWithCurrManager

* The maximum correlation coefficient among the variables in this list is YearsAtCompany and YearsWithCurrManager is 0.7692124.
* The minimum correlation coefficient among the variables in this list is YearsSinceLastPromotion and YearsWithCurrManager is 0.5102236

Correlation exists among the variables in this list.

### 4) Convert continuous variables into Categorical Variables having 5 buckets.

Note: Attrition column in the dataset is your Target Column. EmployeeNumber is the Identifier column.

**Write a function to break the continuous variable into categorical variable having 5 buckets.**

### Function to break the continuous variable into categorical variable having 5 buckets using range to divide.  
  
HRbreaks <- function(x) {  
 HRbreaks\_C <- cut(x, seq(min(x),max(x),(max(x) - min(x))/5))  
 return(HRbreaks\_C)   
}  
  
hrDataTrimmed$AgeC <- HRbreaks(hrDataTrimmed$Age)  
hrDataTrimmed$DailyRateC <- HRbreaks(hrDataTrimmed$DailyRate)  
hrDataTrimmed$HourlyRateC <- HRbreaks(hrDataTrimmed$HourlyRate)  
#  
hrDataTrimmed$MonthlyIncomeC <- HRbreaks(hrDataTrimmed$MonthlyIncome)  
hrDataTrimmed$MonthlyRateC <- HRbreaks(hrDataTrimmed$MonthlyRate)  
hrDataTrimmed$DistanceFromHomeC <- HRbreaks(hrDataTrimmed$DistanceFromHome)  
#  
hrDataTrimmed$NumCompaniesWorkedC <- HRbreaks(hrDataTrimmed$NumCompaniesWorked)  
hrDataTrimmed$PercentSalaryHikeC <- HRbreaks(hrDataTrimmed$PercentSalaryHike)  
hrDataTrimmed$TotalWorkingYearsC <- HRbreaks(hrDataTrimmed$TotalWorkingYears)  
  
#  
hrDataTrimmed$TrainingTimesLastYearC <- HRbreaks(hrDataTrimmed$TrainingTimesLastYear)  
hrDataTrimmed$YearsAtCompanyC <- HRbreaks(hrDataTrimmed$YearsAtCompany)  
hrDataTrimmed$YearsInCurrentRoleC <- HRbreaks(hrDataTrimmed$YearsInCurrentRole)  
#  
hrDataTrimmed$YearsSinceLastPromotionC <- HRbreaks(hrDataTrimmed$YearsSinceLastPromotion)  
hrDataTrimmed$YearsWithCurrManagerC <- HRbreaks(hrDataTrimmed$YearsWithCurrManager)  
  
### Convert integer variables into factors  
  
hrDataTrimmed$Education <- sapply(hrDataTrimmed$Education, function(x) x= as.factor(x))  
hrDataTrimmed$EnvironmentSatisfaction <- sapply(hrDataTrimmed$EnvironmentSatisfaction, function(x) x= as.factor(x))  
hrDataTrimmed$JobInvolvement <- sapply(hrDataTrimmed$JobInvolvement, function(x) x= as.factor(x))  
hrDataTrimmed$JobLevel <- sapply(hrDataTrimmed$JobLevel, function(x) x= as.factor(x))  
hrDataTrimmed$JobSatisfaction <- sapply(hrDataTrimmed$JobSatisfaction, function(x) x= as.factor(x))  
hrDataTrimmed$PerformanceRating <- sapply(hrDataTrimmed$PerformanceRating, function(x) x= as.factor(x))  
hrDataTrimmed$RelationshipSatisfaction <- sapply(hrDataTrimmed$RelationshipSatisfaction, function(x) x= as.factor(x))   
hrDataTrimmed$StockOptionLevel <- sapply(hrDataTrimmed$StockOptionLevel, function(x) x= as.factor(x))   
hrDataTrimmed$WorkLifeBalance <- sapply(hrDataTrimmed$WorkLifeBalance, function(x) x= as.factor(x))  
#

### 5) Compute CHI-SQ statistics for all the independent categorical variable vs Attrition variable

\*\* Write code to return the chi-Square test values for each variable with respect to target variable.\*\*

PrevStatistic <- NA  
i = 0  
for(j in c(1,3:31)){  
 i = i + 1  
 Statistic <- data.frame("Sr No"= i,  
 "Variable"=colnames(hrDataTrimmed)[j],  
 "Chi.Square"=(chisq.test(hrDataTrimmed[ ,2], hrDataTrimmed[ ,j])$statistic),  
 "df"=chisq.test(hrDataTrimmed[ ,2], hrDataTrimmed[ ,j])$parameter,  
 "p.value"=chisq.test(hrDataTrimmed[ ,2],hrDataTrimmed[ ,j])$p.value  
)  
 PrevStatistic = rbind(PrevStatistic,Statistic)  
}

## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect  
  
## Warning in chisq.test(hrDataTrimmed[, 2], hrDataTrimmed[, j]): Chi-squared  
## approximation may be incorrect

#Order the dataframe by descending order of signifigance (by p value)  
PrevStatisticStatistic <-PrevStatistic[order(PrevStatistic$p.value),]  
print(PrevStatisticStatistic,row.names=FALSE)

## Sr.No Variable Chi.Square df p.value  
## 17 MonthlyRate 2.809359e+03 1426 5.798679e-93  
## 16 MonthlyIncome 2.638048e+03 1348 4.049850e-86  
## 3 DailyRate 1.743037e+03 885 1.677505e-58  
## 19 OverTime 1.766051e+02 1 2.671215e-40  
## 13 JobRole 1.723805e+02 8 4.087650e-33  
## 24 TotalWorkingYears 2.446045e+02 39 1.348919e-31  
## 12 JobLevel 1.450580e+02 4 2.330819e-30  
## 1 Age 2.383499e+02 42 2.880537e-29  
## 23 StockOptionLevel 1.211966e+02 3 4.263022e-26  
## 30 YearsWithCurrManager 1.492344e+02 17 3.455982e-23  
## 27 YearsAtCompany 1.907775e+02 36 5.723380e-23  
## 15 MaritalStatus 9.232735e+01 2 8.940669e-21  
## 28 YearsInCurrentRole 1.286019e+02 18 9.802157e-19  
## 11 JobInvolvement 5.698404e+01 3 2.590116e-12  
## 2 BusinessTravel 4.836483e+01 2 3.145656e-11  
## 8 EnvironmentSatisfaction 4.500776e+01 3 9.217620e-10  
## 18 NumCompaniesWorked 5.148881e+01 9 5.646344e-08  
## 14 JobSatisfaction 3.501015e+01 3 1.212247e-07  
## 26 WorkLifeBalance 3.265019e+01 3 3.816994e-07  
## 10 HourlyRate 1.440575e+02 70 4.693401e-07  
## 5 DistanceFromHome 7.633700e+01 28 2.314280e-06  
## 7 EducationField 3.204935e+01 5 5.809061e-06  
## 4 Department 2.159201e+01 2 2.048111e-05  
## 25 TrainingTimesLastYear 3.029290e+01 6 3.457535e-05  
## 29 YearsSinceLastPromotion 4.368994e+01 15 1.229497e-04  
## 22 RelationshipSatisfaction 1.048214e+01 3 1.488257e-02  
## 20 PercentSalaryHike 2.670550e+01 14 2.102204e-02  
## 9 Gender 2.389563e+00 1 1.221477e-01  
## 6 Education 6.147923e+00 4 1.883703e-01  
## 21 PerformanceRating 7.588682e-03 1 9.305817e-01  
## NA <NA> NA NA NA

**Observation**

#### Hypothesis Validation

Let us formulate th null hypothesis and alternative hypothesis.

Null Hypothesis : H0: There is no significant relationship between Attrition and the independent variable

Alternative Hypothesis: HA: There is significant relationship between Attrition and the independent variable

From the above table, we observe that following variables are not significant at 5% level of Significance since p-value > 5 %:

* Gender
* Education
* PerformanceRating

We accept the null hypothesis for the above variables and conclude that there is no significant relationship between attrition and the above mentioned variables.

For the rest of the variables, since p-value < 5%, we reject the null hypothesis and conclude that there is significant relationship between attrition and the above mentioned variables.

#### We shall retain other than the above variables in the data frame, df

df <- hrDataTrimmed[,-c(7,10,22)]  
new\_df <- df[c(2,29,3,30,5,34,7,8,31,10,11,12,13,14,32,33,35,18,36,20,21,37,38,24,39,40,41,42)]

### 6) Based on CHI-SQ statistics find the important variables

From the above table, we observe that following variables are significant at 5% level of Significance and hence **important**:

* Age
* BusinessTravel
* DailyRate
* Department
* DistanceFromHome
* EducationField
* EnvironmentSatisfaction
* HourlyRate
* JobInvolvement"
* JobLevel
* JobRole
* JobSatisfaction
* MaritalStatus
* MonthlyIncome
* MonthlyRate
* NumCompaniesWorked
* OverTime
* PercentSalaryHike
* RelationshipSatisfaction
* StockOptionLevel
* TotalWorkingYears
* TrainingTimesLastYear
* WorkLifeBalance
* YearsAtCompany
* YearsInCurrentRole
* YearsSinceLastPromotion
* YearsWithCurrManager

### 7) Build Model on HR Employee Attrition Data using "CART", "Random Forest" and "Neural Network" w.r.t. your output in step 6

We split the data into training and testing sample data in the ratio 70:30.

set.seed(1234)  
  
### Split data into training and testing data  
  
new\_df$random <- runif(nrow(new\_df), 0, 1);  
new\_df.TRAIN <- new\_df[which(new\_df$random <= 0.7),]  
new\_df.TEST <- new\_df[which(new\_df$random > 0.7),]

### a) CART Model

Recursive partitioning is a fundamental tool in data mining. It helps us explore the structure of a set of data, while developing easy to visualize decision rules for predicting a categorical (classification tree) or continuous (regression tree) outcome.

#### CART Modeling via rpart

Classification and Regression Trees (as described by Brieman, Freidman, Olshenm and Stone) can be generated through the **rpart** package.

##### i) Grow the tree

Control is an optional parameter for controlling tree growth. We use rpart with contol = rpart.control(minsplit=100, minbucket = 10, cp = 0, xval = 10).

* Minimum number of observations in a node to be 100 before attempting a split
* A split must decrease the overall lack of fit by a factor of 0 (Cost complexity factor) before being attempted.
* Minimum number of observations in any terminal node.

#CART Algorithm  
  
r.ctrl = rpart.control(minsplit=100, minbucket = 10, cp = 0, xval = 10)  
  
m1 <- rpart(Attrition ~ AgeC + BusinessTravel  
 + DailyRateC + Department + DistanceFromHomeC + EducationField   
 + EnvironmentSatisfaction + HourlyRateC + JobInvolvement + JobLevel + JobRole  
 + JobSatisfaction + MaritalStatus + MonthlyIncomeC + MonthlyRateC + NumCompaniesWorkedC + OverTime + PercentSalaryHikeC + RelationshipSatisfaction + StockOptionLevel + TotalWorkingYearsC + TrainingTimesLastYearC + WorkLifeBalance + YearsAtCompanyC + YearsInCurrentRoleC + YearsSinceLastPromotionC + YearsWithCurrManagerC, data=new\_df.TRAIN, method = "class", control = r.ctrl)

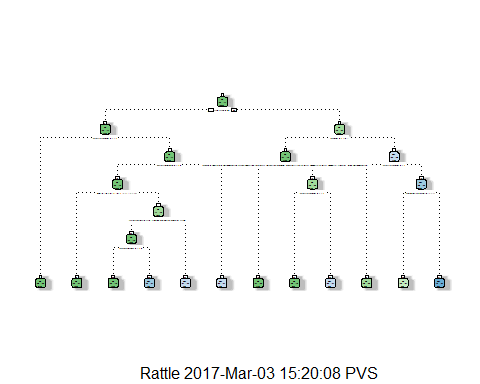
m1

## n= 2082   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 2082 348 No (0.83285303 0.16714697)   
## 2) OverTime=No 1495 159 No (0.89364548 0.10635452)   
## 4) StockOptionLevel=1,3,2 853 52 No (0.93903869 0.06096131) \*  
## 5) StockOptionLevel=0 642 107 No (0.83333333 0.16666667)   
## 10) WorkLifeBalance=3,2,4 605 88 No (0.85454545 0.14545455)   
## 20) AgeC=(34.8,43.2],(43.2,51.6],(51.6,60] 305 20 No (0.93442623 0.06557377) \*  
## 21) AgeC=(18,26.4],(26.4,34.8] 300 68 No (0.77333333 0.22666667)   
## 42) EducationField=Life Sciences,Marketing,Medical,Other 262 48 No (0.81679389 0.18320611)   
## 84) JobInvolvement=3,2,4 244 37 No (0.84836066 0.15163934) \*  
## 85) JobInvolvement=1 18 7 Yes (0.38888889 0.61111111) \*  
## 43) EducationField=Human Resources,Technical Degree 38 18 Yes (0.47368421 0.52631579) \*  
## 11) WorkLifeBalance=1 37 18 Yes (0.48648649 0.51351351) \*  
## 3) OverTime=Yes 587 189 No (0.67802385 0.32197615)   
## 6) JobLevel=2,3,4,5 360 67 No (0.81388889 0.18611111)   
## 12) JobRole=Healthcare Representative,Human Resources,Laboratory Technician,Manager,Manufacturing Director,Research Director,Research Scientist,Sales Representative 221 19 No (0.91402715 0.08597285) \*  
## 13) JobRole=Sales Executive 139 48 No (0.65467626 0.34532374)   
## 26) StockOptionLevel=1,3,2 76 12 No (0.84210526 0.15789474) \*  
## 27) StockOptionLevel=0 63 27 Yes (0.42857143 0.57142857) \*  
## 7) JobLevel=1 227 105 Yes (0.46255507 0.53744493)   
## 14) StockOptionLevel=1,2 96 35 No (0.63541667 0.36458333) \*  
## 15) StockOptionLevel=0,3 131 44 Yes (0.33587786 0.66412214)   
## 30) DistanceFromHomeC=(1,6.6] 58 27 No (0.53448276 0.46551724) \*  
## 31) DistanceFromHomeC=(6.6,12.2],(12.2,17.8],(17.8,23.4],(23.4,29] 73 13 Yes (0.17808219 0.82191781) \*

##### ii) Examine the results

We display cp table and plot cross-validation results.

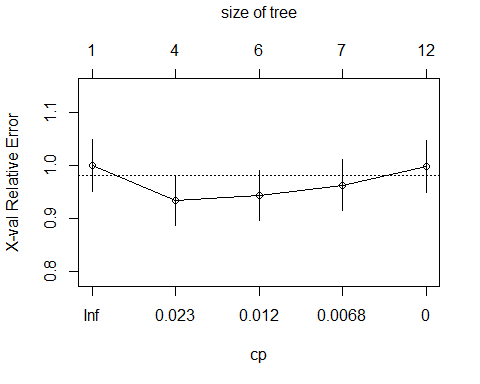
fancyRpartPlot(m1)



printcp(m1)

##   
## Classification tree:  
## rpart(formula = Attrition ~ AgeC + BusinessTravel + DailyRateC +   
## Department + DistanceFromHomeC + EducationField + EnvironmentSatisfaction +   
## HourlyRateC + JobInvolvement + JobLevel + JobRole + JobSatisfaction +   
## MaritalStatus + MonthlyIncomeC + MonthlyRateC + NumCompaniesWorkedC +   
## OverTime + PercentSalaryHikeC + RelationshipSatisfaction +   
## StockOptionLevel + TotalWorkingYearsC + TrainingTimesLastYearC +   
## WorkLifeBalance + YearsAtCompanyC + YearsInCurrentRoleC +   
## YearsSinceLastPromotionC + YearsWithCurrManagerC, data = new\_df.TRAIN,   
## method = "class", control = r.ctrl)  
##   
## Variables actually used in tree construction:  
## [1] AgeC DistanceFromHomeC EducationField JobInvolvement   
## [5] JobLevel JobRole OverTime StockOptionLevel   
## [9] WorkLifeBalance   
##   
## Root node error: 348/2082 = 0.16715  
##   
## n= 2082   
##   
## CP nsplit rel error xerror xstd  
## 1 0.041188 0 1.00000 1.00000 0.048921  
## 2 0.012931 3 0.87644 0.93391 0.047589  
## 3 0.011494 5 0.85057 0.94253 0.047767  
## 4 0.004023 6 0.83908 0.96264 0.048178  
## 5 0.000000 11 0.81897 0.99713 0.048865

plotcp(m1)



1. Prune tree

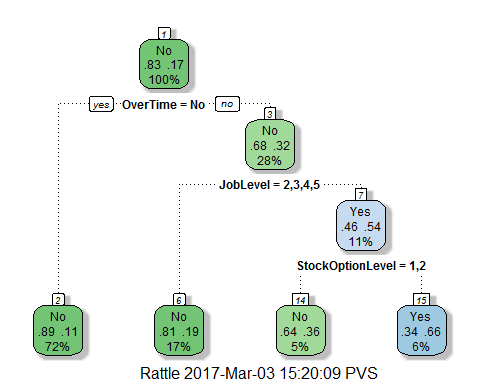
Prune back the tree to avoid overfitting the data. Typically, you want to select a tree size that minimized the cross-validation error, the **xerror** column printed by **printcp**.

We have written the code to automatically select the compexity parameter associated with the smallest cross-validation error.

## Pruning Code  
ptree<- prune(m1,cp= m1$cptable[which.min(m1$cptable[,"xerror"]),"CP"])  
printcp(ptree)

##   
## Classification tree:  
## rpart(formula = Attrition ~ AgeC + BusinessTravel + DailyRateC +   
## Department + DistanceFromHomeC + EducationField + EnvironmentSatisfaction +   
## HourlyRateC + JobInvolvement + JobLevel + JobRole + JobSatisfaction +   
## MaritalStatus + MonthlyIncomeC + MonthlyRateC + NumCompaniesWorkedC +   
## OverTime + PercentSalaryHikeC + RelationshipSatisfaction +   
## StockOptionLevel + TotalWorkingYearsC + TrainingTimesLastYearC +   
## WorkLifeBalance + YearsAtCompanyC + YearsInCurrentRoleC +   
## YearsSinceLastPromotionC + YearsWithCurrManagerC, data = new\_df.TRAIN,   
## method = "class", control = r.ctrl)  
##   
## Variables actually used in tree construction:  
## [1] JobLevel OverTime StockOptionLevel  
##   
## Root node error: 348/2082 = 0.16715  
##   
## n= 2082   
##   
## CP nsplit rel error xerror xstd  
## 1 0.041188 0 1.00000 1.00000 0.048921  
## 2 0.012931 3 0.87644 0.93391 0.047589

fancyRpartPlot(ptree, uniform=TRUE)



**Interpretation**

The tree starts will all of the data at the root node and scans all of the variables for the best one to split on. The root node, at the top, shows 17% of employees leave while 83% do not leave. The 100% of observations reside in this node.

**Node 2:**

* Employees with OverTime = Yes, only 11% leave and 89% stay. 72% of observations reside in this node. This is a terminal node.

**Node 3:**

* Employees with OverTime = No, only 32% leave and 68% stay. 28% of observations reside in this node.

**Node 6:**

\*Employees with JobLevel = 2,3,4,5 is True, only 19% leave and 81% stay. 17% of observations reside in this node. This is a terminal node.

**Node 7:**

* Employees with JobLevel = 2,3,4,5 is False, only 46% leave and 54% stay. 11% of observations reside in this node.

**Node 14:**

* Employees with StockOptionLevel = 1,2 is True, only 36% leave and 64% stay. 5% of observations reside in this node. This is a terminal node.

**Node 15:**

* Employees with StockOptionLevel = 1,2 is False, only 34% leave and 66% stay. 6% of observations reside in this node. This is a terminal node.

**Observation**

The following group of employees leave: \* Employees with OverTime = No \* Employees with OverTime = No and JobLevel is not 2,3,4,5

##### Model performance measures

* K-S statistics looks at maximum differences between distribution of cumulative events and cumulative non-events. K-S is a measure of the degree of separation between the positive and negative distributions. The K-S is 100% if the scores partition the population into two separate groups in which one group contains all the positives and the other all the negatives.

##### A few thumbs of rules for K-S

|  |  |
| --- | --- |
| Values Range | Remarks |
| 0.40 - 0.70 | Good |

* Area under the ROC curve (AUC - ROC) is the ratio under the curve and the total area. ROC curve is the plot between sensitivity (a.k.a. True Positive Rate) and (1 - specificity) (a.k.a False Positive Rate)

##### A few thumbs of rules for AUC - ROC

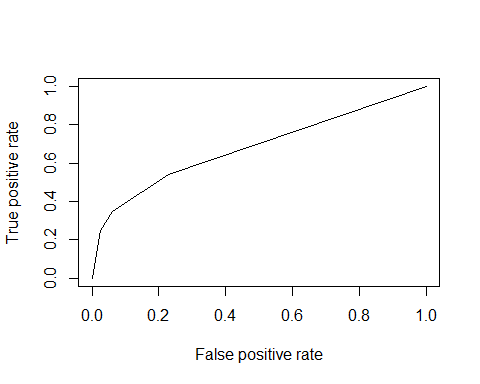
|  |  |
| --- | --- |
| Values Range | Remarks |
| 0.90 - 1 | Excellent |
| 0.80 - 0.90 | Good |
| 0.70 - 0.80 | Fair |
| 0.60 - 0.70 | Poor |
| 0.50 - 0.60 | Fail |

* Gini is the ratio between area between the ROC curve and the diagonal line & the area of the above triangle.

##### A few thumbs of rules for Gini coefficient

|  |  |
| --- | --- |
| Values Range | Remarks |
| Above 60% | Good |

new\_df.TRAIN$predict.class <- predict(ptree, new\_df.TRAIN, type="class")  
new\_df.TRAIN$predict.score <- predict(ptree,new\_df.TRAIN)  
  
pred <- prediction(new\_df.TRAIN$predict.score[,2], new\_df.TRAIN$Attrition)  
perf <- performance(pred, "tpr", "fpr")  
  
### Plot   
  
plot(perf)



KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])  
auc <- performance(pred,"auc");   
auc <- as.numeric(auc@y.values)  
gini = ineq(new\_df.TRAIN$predict.score[,2], type="Gini")  
  
with(new\_df.TRAIN, table(Attrition, predict.class))

## predict.class  
## Attrition No Yes  
## No 1690 44  
## Yes 261 87

auc

## [1] 0.6836992

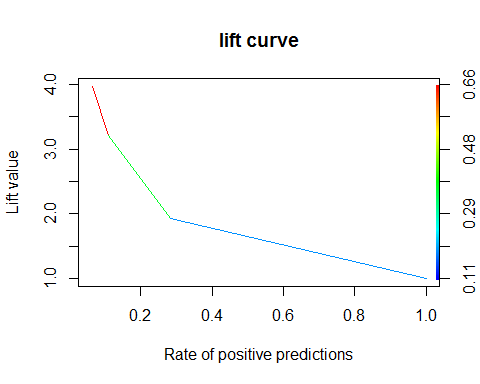
KS

## [1] 0.3135763

gini

## [1] 0.3059889

### Function for Gain and Lift chart  
  
lift <- function(depvar, predcol, groups=10) {  
if(is.factor(depvar)) depvar <- as.integer(as.character(depvar))  
if(is.factor(predcol)) predcol <- as.integer(as.character(predcol))  
helper = data.frame(cbind(depvar, predcol))  
helper[,"bucket"] = ntile(-helper[,"predcol"], groups)  
gaintable = helper %>% group\_by(bucket) %>%  
 summarise\_at(vars(depvar), funs(total = n(),  
 totalresp=sum(., na.rm = TRUE))) %>%  
 mutate(Cumresp = cumsum(totalresp),  
 Gain=Cumresp/sum(totalresp)\*100,  
 Cumlift=Gain/(bucket\*(100/groups)))  
return(gaintable)  
}  
  
# Lift chart  
  
perf <- performance(pred,"lift","rpp")  
plot(perf, main="lift curve", colorize=T)

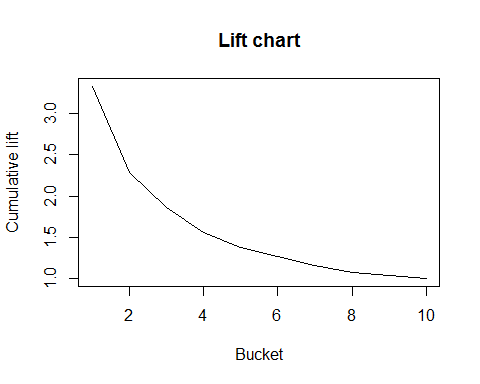


target <- ifelse(new\_df.TRAIN$Attrition == "Yes",1,0)  
dt <- lift(target, new\_df.TRAIN$predict.score[,2], groups = 10)

### Print the Gain and Lift table  
  
print(dt)

## # A tibble: 10 × 6  
## bucket total totalresp Cumresp Gain Cumlift  
## <dbl> <int> <dbl> <dbl> <dbl> <dbl>  
## 1 1 209 116 116 33.33333 3.333333  
## 2 2 208 43 159 45.68966 2.284483  
## 3 3 208 35 194 55.74713 1.858238  
## 4 4 208 23 217 62.35632 1.558908  
## 5 5 208 23 240 68.96552 1.379310  
## 6 6 209 25 265 76.14943 1.269157  
## 7 7 208 17 282 81.03448 1.157635  
## 8 8 208 17 299 85.91954 1.073994  
## 9 9 208 27 326 93.67816 1.040868  
## 10 10 208 22 348 100.00000 1.000000

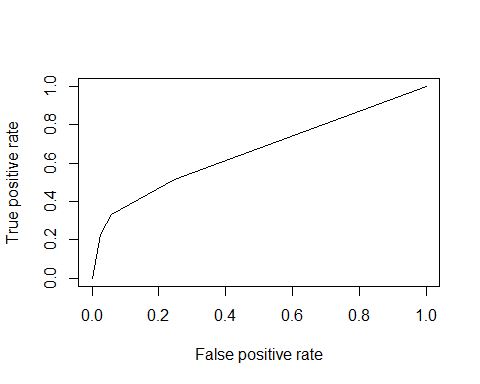
### Lift Chart  
  
plot(dt$bucket, dt$Cumlift, type="l", main = "Lift chart", ylab="Cumulative lift", xlab="Bucket")



#### Model performance using Testing sample data

new\_df.TEST$predict.class <- predict(ptree, new\_df.TEST, type="class")  
new\_df.TEST$predict.score <- predict(ptree,new\_df.TEST)  
  
pred <- prediction(new\_df.TEST$predict.score[,2], new\_df.TEST$Attrition)  
perf <- performance(pred, "tpr", "fpr")

### Plot   
plot(perf)



KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])  
auc <- performance(pred,"auc");   
auc <- as.numeric(auc@y.values)  
gini = ineq(new\_df.TEST$predict.score[,2], type="Gini")  
  
with(new\_df.TEST, table(Attrition, predict.class))

## predict.class  
## Attrition No Yes  
## No 714 18  
## Yes 97 29

auc

## [1] 0.6634791

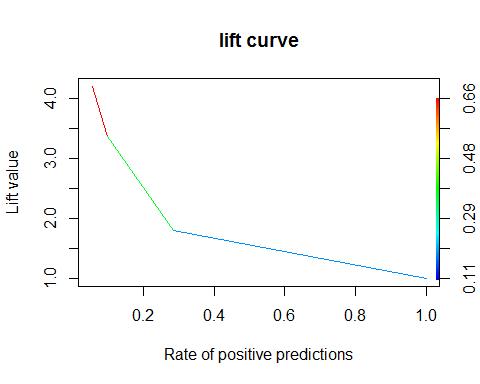
KS

## [1] 0.2745902

gini

## [1] 0.2923077

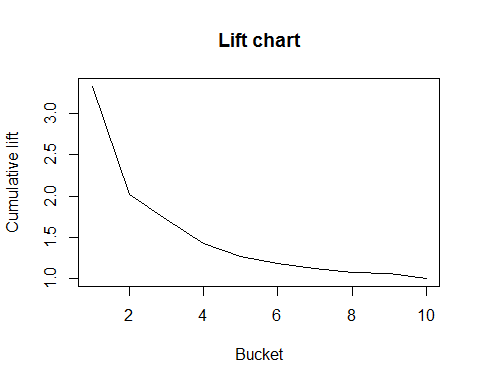
# A lift chart  
  
perf <- performance(pred, measure="lift", x.measure="rpp")  
plot(perf, main="lift curve", colorize=T)



###  
  
target <- ifelse(new\_df.TEST$Attrition == "Yes",1,0)  
dt <- lift(target, new\_df.TEST$predict.score[,2], groups = 10)  
  
### Print the Gain and Lift table  
  
print(dt)

## # A tibble: 10 × 6  
## bucket total totalresp Cumresp Gain Cumlift  
## <dbl> <int> <dbl> <dbl> <dbl> <dbl>  
## 1 1 86 42 42 33.33333 3.333333  
## 2 2 86 9 51 40.47619 2.023810  
## 3 3 86 14 65 51.58730 1.719577  
## 4 4 86 7 72 57.14286 1.428571  
## 5 5 85 8 80 63.49206 1.269841  
## 6 6 86 9 89 70.63492 1.177249  
## 7 7 86 10 99 78.57143 1.122449  
## 8 8 86 9 108 85.71429 1.071429  
## 9 9 86 12 120 95.23810 1.058201  
## 10 10 85 6 126 100.00000 1.000000

### Lift Chart  
  
plot(dt$bucket, dt$Cumlift, type="l", main = "Lift chart", ylab="Cumulative lift", xlab="Bucket")



**Observation**

|  |  |  |  |
| --- | --- | --- | --- |
| Statistic Name | Value for Training data | Value for Testing data | Remarks |
| K-S statistics | 0.31 | 0.27 | Modest fit - value remains almost same in both samples |
| AUC | 0.68 | 0.66 | Modest fit - value remains almost same in both samples |
| gini | 0.31 | 0.29 | Modest fit - value remains almost same in both samples |

**Interpretation of Cumulative Lift**

The cumulative lift of 2.28 for training sample data and 2.02 for testing sample data for top two deciles, means that when selecting 20% of the records based on the model, one can expect 2.28 (for training data) or 2.02 (for testing data) times the total number of targets (events) found by randomly selecting 20% -of-records without a model.

In terms of employee attrition model, we can say we can cover 2.28 (for training data) or 2.02 times (for testing data) the number of attritors by selecting only 20% of the customers based on the model as compared to 20% employee selection randomly.

**Interpretation of Gain**

Here, we talk about % of targets (events) covered at a given decile level. For example, 55.7% (for training data) or 51.6% (for testing data) of events covered in top 30% of data based on model.

**Model Performance for CART model**

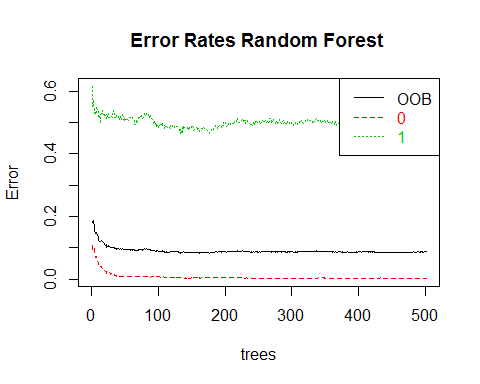
|  |  |  |
| --- | --- | --- |
| **Performance Metric** | **Training Data** | **Testing Data** |
| Accuracy Classification score | **0.86** | **0.88** |
| Area Under the Receiver Operating Characteristic Curve | **0.68** | **0.66** |
| Gini Coefficient | **0.31** | **0.29** |

#### b) Build model based on Random Forest

RF <- randomForest(Attrition ~ ., data = Rdev[,-32],   
 ntree=501, mtry = 5, nodesize = 10,  
 importance=TRUE)  
print(RF)

##   
## Call:  
## randomForest(formula = Attrition ~ ., data = Rdev[, -32], ntree = 501, mtry = 5, nodesize = 10, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 501  
## No. of variables tried at each split: 5  
##   
## OOB estimate of error rate: 8.75%  
## Confusion matrix:  
## No Yes class.error  
## No 1702 6 0.003512881  
## Yes 174 176 0.497142857

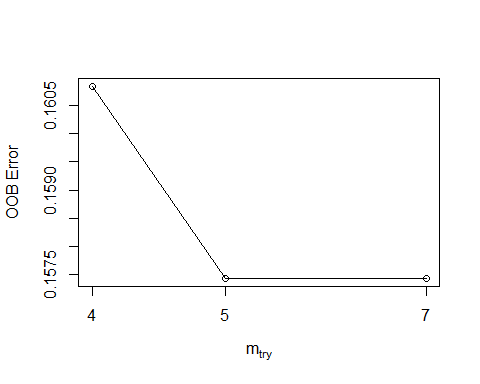
#### Error Rates for Random Forest



#### Tune Random Forest

tRF <- tuneRF(x = Rdev[,-c(2,32)],   
 y=Rdev$Attrition,  
 mtryStart = 5,   
 ntreeTry=100,   
 stepFactor = 1.5,   
 improve = 0.001,   
 trace=TRUE,   
 plot = TRUE,  
 doBest = TRUE,  
 nodesize = 150,   
 importance=TRUE  
)

## mtry = 5 OOB error = 15.74%   
## Searching left ...  
## mtry = 4 OOB error = 16.08%   
## -0.02160494 0.001   
## Searching right ...  
## mtry = 7 OOB error = 15.74%   
## 0 0.001



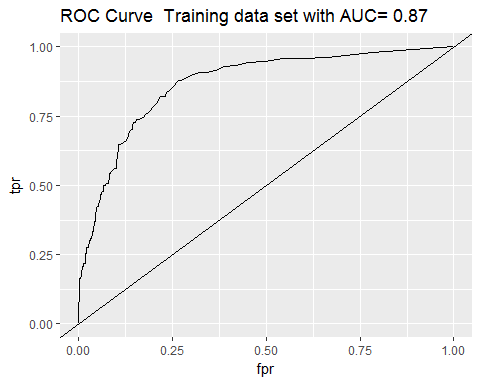
print(tRF)

##   
## Call:  
## randomForest(x = x, y = y, mtry = res[which.min(res[, 2]), 1], nodesize = 150, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 5  
##   
## OOB estimate of error rate: 15.89%

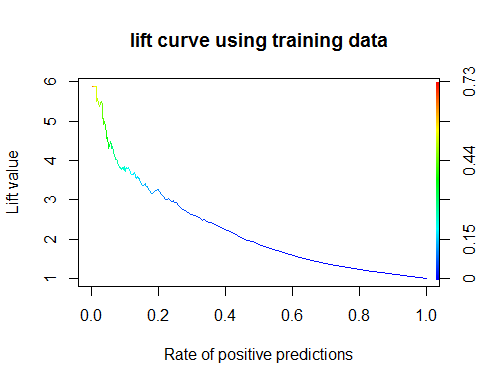
#### Model Performance measures using Training data set

## Confusion matrix:  
## No Yes class.error  
## No 1704 4 0.00234192  
## Yes 323 27 0.92285714

tRF.pr = predict(tRF,type="prob",newdata=Rdev)[,2]  
tRF.pred = prediction(tRF.pr,Rdev$Attrition)  
tRF.perf = performance(tRF.pred,measure = "tpr", x.measure="fpr")  
auc <- performance(tRF.pred, measure="auc")  
auc <- auc@y.values[[1]]  
roc.data <- data.frame(fpr=unlist(tRF.perf@x.values), tpr=unlist(tRF.perf@y.values),model="GLM")  
ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr))+  
geom\_line(aes(y=tpr)) + geom\_abline() +   
ggtitle(paste0("ROC Curve Training data set with AUC= ",round(auc,2)))



###  
  
tRF.perf = performance(tRF.pred,measure = "lift", x.measure="rpp")  
plot(tRF.perf, main="lift curve using training data", colorize=T)



gini

## [1] 0.7480058

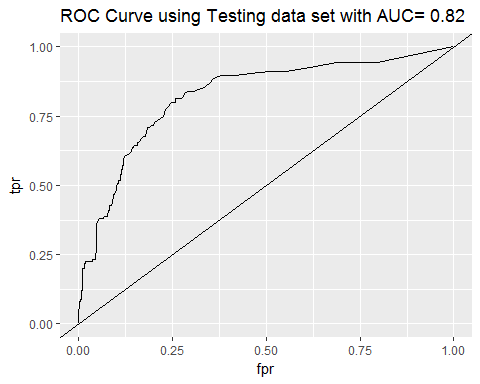
#### Model Performance measures using Testing data set

## Scoring Holdout sample  
  
Rhold$predict.class <- predict(tRF, Rhold[,-32], type="class")  
Rhold$predict.score <- predict(tRF, Rhold[,-32], type="prob")  
  
with(Rhold, table(Attrition, predict.class))

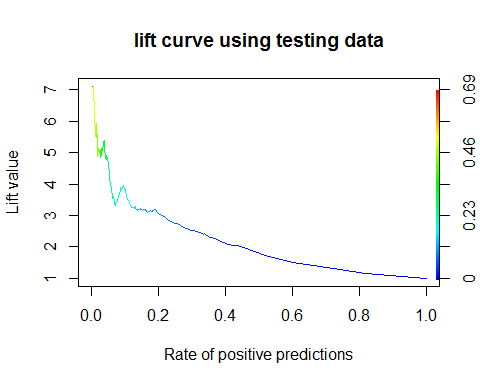
## Confusion matrix:

## predict.class  
## Attrition No Yes  
## No 758 0  
## Yes 118 6

tRF.pr = predict(tRF,type="prob",newdata=Rhold)[,2]  
tRF.pred = prediction(tRF.pr,Rhold$Attrition)  
tRF.perf = performance(tRF.pred,"tpr","fpr")  
  
auc <- performance(tRF.pred, measure="auc")  
auc <- auc@y.values[[1]]  
roc.data <- data.frame(fpr=unlist(tRF.perf@x.values), tpr=unlist(tRF.perf@y.values),model="GLM")  
ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr))+  
geom\_line(aes(y=tpr)) + geom\_abline() +  
ggtitle(paste0("ROC Curve using Testing data set with AUC= ",round(auc,2)))



###  
  
tRF.perf = performance(tRF.pred,measure = "lift", x.measure="rpp")  
plot(tRF.perf, main="lift curve using testing data", colorize=T)



gini

## [1] 0.7480058

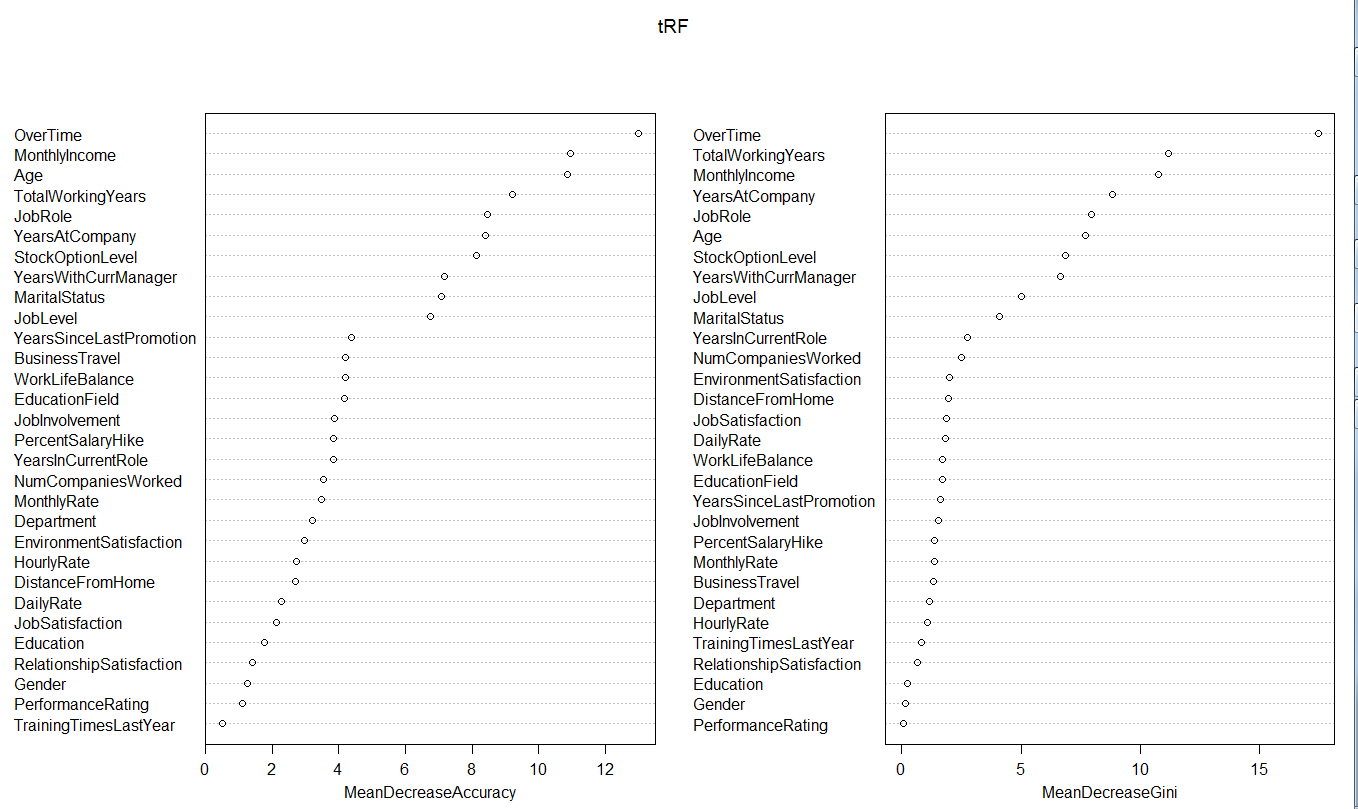
**Model Performance for Random Forest model**

|  |  |  |
| --- | --- | --- |
| **Performance Metric** | **Training Data** | **Testing Data** |
| Accuracy Classification score | **0.84** | **0.87** |
| Area Under the Receiver Operating Characteristic Curve | **0.87** | **0.82** |
| Gini Coefficient | **0.75** | **0.75** |

#### Variable importance  
importance(tRF)

## No Yes MeanDecreaseAccuracy  
## Age 8.0309082 7.8615120 10.8513307  
## BusinessTravel 3.6409335 3.6340138 4.1981894  
## DailyRate 1.9685060 1.7281504 2.2788971  
## Department 2.6334405 1.4833258 3.1929296  
## DistanceFromHome 2.4734989 1.6161281 2.6967079  
## Education 1.9007952 -0.4783219 1.7529880  
## EducationField 3.4731245 3.1568844 4.1503252  
## EnvironmentSatisfaction 2.4121462 3.0504883 2.9597603  
## Gender 0.7032614 1.6423739 1.2388764  
## HourlyRate 2.0518903 2.0866043 2.7379306  
## JobInvolvement 2.1164113 3.8831638 3.8630322  
## JobLevel 4.1157261 6.1707461 6.7565077  
## JobRole 3.6146193 7.8164529 8.4626260  
## JobSatisfaction 2.4192874 0.9708640 2.1252057  
## MaritalStatus 5.2449518 6.0742965 7.0850556  
## MonthlyIncome 6.6941451 9.1962697 10.9638627  
## MonthlyRate 2.2581351 3.8328759 3.4674560  
## NumCompaniesWorked 3.4820448 1.9451697 3.5402772  
## OverTime 11.5718037 13.1553323 12.9998636  
## PercentSalaryHike 3.1920228 3.2965730 3.8384838  
## PerformanceRating 0.6302658 0.6702553 1.0972141  
## RelationshipSatisfaction 1.2929031 -0.1023281 1.4036535  
## StockOptionLevel 5.2035176 7.8694958 8.1253750  
## TotalWorkingYears 5.0196945 8.8825147 9.2093440  
## TrainingTimesLastYear 0.2289194 0.5904748 0.4994569  
## WorkLifeBalance 3.1954374 3.5163098 4.1976155  
## YearsAtCompany 2.8544931 8.1940329 8.4105121  
## YearsInCurrentRole 0.8913665 4.0610209 3.8254955  
## YearsSinceLastPromotion 3.0463144 2.2809783 4.3642978  
## YearsWithCurrManager 1.1677901 6.6094278 7.1712995  
## MeanDecreaseGini  
## Age 7.7016260  
## BusinessTravel 1.3341879  
## DailyRate 1.8301479  
## Department 1.1469752  
## DistanceFromHome 1.9466421  
## Education 0.2160060  
## EducationField 1.7126339  
## EnvironmentSatisfaction 1.9799825  
## Gender 0.1322197  
## HourlyRate 1.0858849  
## JobInvolvement 1.5423379  
## JobLevel 5.0194864  
## JobRole 7.9500261  
## JobSatisfaction 1.8564478  
## MaritalStatus 4.0839182  
## MonthlyIncome 10.7734336  
## MonthlyRate 1.3689956  
## NumCompaniesWorked 2.5155593  
## OverTime 17.4545958  
## PercentSalaryHike 1.3762878  
## PerformanceRating 0.0535952  
## RelationshipSatisfaction 0.6696879  
## StockOptionLevel 6.8656469  
## TotalWorkingYears 11.1862205  
## TrainingTimesLastYear 0.8232844  
## WorkLifeBalance 1.7152841  
## YearsAtCompany 8.8509766  
## YearsInCurrentRole 2.7335278  
## YearsSinceLastPromotion 1.6107190  
## YearsWithCurrManager 6.6647739

varImpPlot(tRF)



We observe that from the above plots, top ten important variables for predicting the HR attrition are:

|  |  |
| --- | --- |
| 1) OverTime | 6) StockOptionLevel |
| 2) TotalWorking Years | 7) YearsWithCurrentManager |
| 3) MonthlyIncome | 8) JobLevel |
| 4) YearsAtCompany | 9) MaritalStatus |
| 5) Age | 10) YearsInCurrentRole |

#### c) Build model based on Artificial Neural Network

#Neural network modelling  
library(neuralnet)

## Warning: package 'neuralnet' was built under R version 3.2.5

names(nndev)

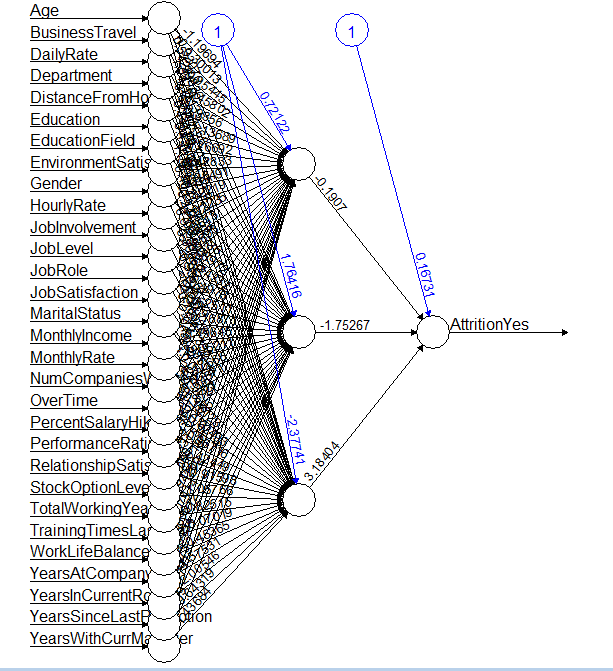
## [1] "Age" "BusinessTravel"   
## [3] "DailyRate" "Department"   
## [5] "DistanceFromHome" "Education"   
## [7] "EducationField" "EnvironmentSatisfaction"   
## [9] "Gender" "HourlyRate"   
## [11] "JobInvolvement" "JobLevel"   
## [13] "JobRole" "JobSatisfaction"   
## [15] "MaritalStatus" "MonthlyIncome"   
## [17] "MonthlyRate" "NumCompaniesWorked"   
## [19] "OverTime" "PercentSalaryHike"   
## [21] "PerformanceRating" "RelationshipSatisfaction"  
## [23] "StockOptionLevel" "TotalWorkingYears"   
## [25] "TrainingTimesLastYear" "WorkLifeBalance"   
## [27] "YearsAtCompany" "YearsInCurrentRole"   
## [29] "YearsSinceLastPromotion" "YearsWithCurrManager"   
## [31] "AttritionYes"

n <- names(nndev)  
f <- as.formula(paste("AttritionYes ~",   
 paste(n[!n %in% "AttritionYes"], collapse = " + ")))  
nn1 <- neuralnet(formula = f,  
 data = nndev,   
 hidden = 3,  
 err.fct = "sse",  
 linear.output = FALSE,  
 lifesign = "full",  
 lifesign.step = 10,  
 stepmax = 2000  
 ##startweights = startweightsObj  
)

## hidden: 3 thresh: 0.01 rep: 1/1 steps: 10 min thresh: 112.462807  
## 20 min thresh: 0.2386431647  
## 30 error: 145.2381 time: 0.11 secs

#### Visualize the Neural Net

***The black lines represent the weighted vectors between the neurons. The blue line represents the bias added.***



#### Let us see the distribution of the estimated results.

quantile(nn1$net.result[[1]], c(0,1,5,10,25,50,75,90,95,99,100)/100)

## 0% 1% 5% 10% 25%   
## 0.1700380303 0.1700380303 0.1700380303 0.1700380303 0.1700380303   
## 50% 75% 90% 95% 99%   
## 0.1700380303 0.1700380303 0.1700380303 0.1700380303 0.1700380303   
## 100%   
## 0.1700380303

**There is no difference in the estimated values distributed across various quantiles shown above.**

#### Let us scale the variables and see if it has an impact on neural network model output.

n1 <- names(nndev[,-31])  
x<-subset(nndev,select=n1)  
nndevscaled <- scale(x)  
nndevscaled <- cbind(nndev[31], nndevscaled)

### Build the model after scaling

nn2 <- neuralnet(formula = f,  
 data = nndevscaled,   
 hidden = 6,  
 err.fct = "sse",  
 linear.output = FALSE,  
 lifesign = "full",  
 lifesign.step = 1,  
 stepmax = 4000  
 )

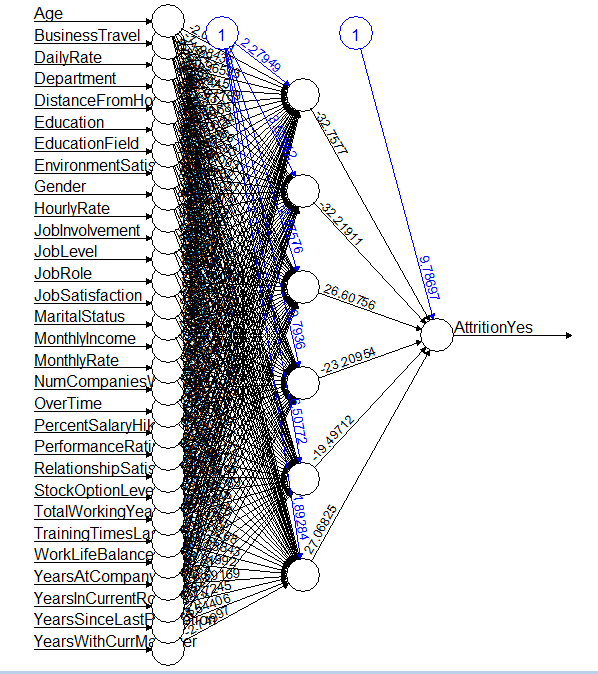
## hidden: 6 thresh: 0.01 rep: 1/1 steps: 1 min thresh: 120.8356429  
## 2 min thresh: 73.68070705  
## 3 min thresh:

...

...

## 3807 min thresh: 0.01219062235  
## 3808 min thresh: 0.01219062235  
## 3809 error: 31.46826 time: 20.58 secs

### View the plot of the model



### Observe the distribution of the predicted probabilities

quantile(nn2$net.result[[1]], c(0,1,5,10,25,50,75,90,95,99,100)/100)

## 0%   
## 0.0000000000000000000000000000000000000000003048669989   
## 1%   
## 0.0000000000000000000000000000000000000000023780197816   
## 5%   
## 0.0000000000000000000000000000000045822515536851619608   
## 10%   
## 0.0000000000000000000000000000011510777618284833282488   
## 25%   
## 0.0000000000000000000027171215696293867564083818200160   
## 50%   
## 0.0000000000002156433949568255330469374042579033812217   
## 75%   
## 0.0000627312890574029384960913491298128974449355155230   
## 90%   
## 0.9905117931013973642251357887289486825466156005859375   
## 95%   
## 0.9999927998420107266852596694661770015954971313476562   
## 99%   
## 0.9999999999998616662111317054950632154941558837890625   
## 100%   
## 1.0000000000000000000000000000000000000000000000000000

**There is a wide difference in the estimated values distributed across various quantiles shown above.**

#### Performance measures using Training data

misClassTable = data.frame(Target = nndevscaled$Attrition, Predict.score = nn2$net.result[[1]] )  
misClassTable$Predict.class = ifelse(misClassTable$Predict.score>0.8,1,0)  
with(misClassTable, table(Target, Predict.class))

**Confusion Matrix**

## Predict.class  
## Target 0 1  
## 0 1706 2  
## 1 61 289

y\_predict <- misClassTable$Predict.class  
y\_true <- nndevscaled$Attrition  
  
### Accuracy Classification score  
  
Accuracy(y\_predict, y\_true)

## [1] 0.9693877551

### Calculate Area Under the Receiver Operating Characteristic Curve  
  
AUC(y\_pred =misClassTable$Predict.class, y\_true= nndevscaled$Attrition)

## [1] 0.9122716628

### Calculate Gini Coefficient  
  
Gini(y\_predict, y\_true)

## [1] 0.812763466

### Compute Kolmogorov - Smirnov Statistic  
  
KS\_Stat(y\_predict, y\_true)

## [1] 82.45433255

### Compute Are Under the Lift Chart from prediction scores  
  
LiftAUC(y\_pred =misClassTable$Predict.class, y\_true= nndevscaled$Attrition)

## [1] 2.936236963

### Observe the distribution of the predicted probabilities for testing data

quantile(nnhold$Predict.score, c(0,1,5,10,25,50,75,90,95,99,100)/100)

## 0%   
## 0.0000000000000000000000000000000000000000003048842729   
## 1%   
## 0.0000000000000000000000000000000000000000015800697688   
## 5%   
## 0.0000000000000000000000000000000014897985581493069581   
## 10%   
## 0.0000000000000000000000000000002686906718349676023282   
## 25%   
## 0.0000000000000000000003097797658694537535975510600395   
## 50%   
## 0.0000000000000603080942087114665146538317053170885629   
## 75%   
## 0.0000059940977959502732093019505832387494592694565654   
## 90%   
## 0.9879715864843254813365547306602820754051208496093750   
## 95%   
## 0.9999872261196789402504236932145431637763977050781250   
## 99%   
## 0.9999999999999762412272730216500349342823028564453125   
## 100%   
## 1.0000000000000000000000000000000000000000000000000000

#### Performance measures using Testing data

x <- subset(nnhold,   
 select = n1)  
x.scaled <- scale(x)  
compute.output = compute(nn2, x.scaled)  
nnhold$Predict.score = compute.output$net.result

#### Calculate required metrics

nnhold$Predict.class = ifelse(nnhold$Predict.score>0.8,1,0)  
with(nnhold, table(AttritionYes, Predict.class))

**Confusion Matrix**

## Predict.class  
## AttritionYes 0 1  
## 0 731 27  
## 1 42 82

y\_predict <- nnhold$Predict.class  
y\_true <- nnhold$AttritionYes  
  
### Accuracy Classification score  
  
Accuracy(y\_predict, y\_true)

## [1] 0.9217687075

### Calculate Area Under the Receiver Operating Characteristic Curve  
  
AUC(y\_pred =nnhold$Predict.class, y\_true= nnhold$AttritionYes)

## [1] 0.8128351349

### Calculate Gini Coefficient  
  
Gini(y\_predict, y\_true)

## [1] 0.5855391948

### Compute Kolmogorov - Smirnov Statistic  
  
KS\_Stat(y\_predict, y\_true)

## [1] 62.56702698

### Compute Area Under the Lift Chart from prediction scores  
  
LiftAUC(y\_pred =nnhold$Predict.class, y\_true= nnhold$AttritionYes)

## [1] 2.783059164

**Model Performance for ANN model**

|  |  |  |
| --- | --- | --- |
| **Performance Metric** | **Training Data** | **Testing Data** |
| Accuracy Classification score | **0.97** | **0.92** |
| Area Under the Receiver Operating Characteristic Curve | **0.91** | **0.81** |
| Gini Coefficient | **0.81** | **0.59** |
| Kolmogorov - Smirnov Statistic | **82.45%** | **62.57%** |
| Area Under the Lift Chart from prediction scores | **2.94** | **2.78** |

**Compare model performance**

We shall compare the performance metrics for various models we have built:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Performance Metric** | **CART** | | **Random Forest** | | **Neural Network** | |
| **Training Data** | **Testing Data** | **Training Data** | **Testing Data** | **Training Data** | **Testing Data** |
| Accuracy Classification score | **0.86** | **0.88** | **0.84** | **0.87** | **0.97** | **0.92** |
| Area Under the Receiver Operating Characteristic Curve | **0.68** | **0.66** | **0.87** | **0.82** | **0.81** | **0.59** |
| Gini Coefficient | **0.31** | **0.29** | **0.75** | **0.75** | **0.81** | **0.59** |

**We observe that Random Forest model performs better than other models, namely, CART and Neural Network.**