IBM Human Resources

IBM Human Resources

Our Objective is to determine which factors and to what degree each factor is driving employee attrition. We get the data from IBM Data Science Team Kaggle.com - HR Analytics All the libraries we used:

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```
library(MASS)
library(tree)
library(knitr)
library(kableExtra)
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

```
library(gbm)
```

```
## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3
```

```r
library(e1071)
```

Data Cleaning

```r
attribution = read.csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
str(attribution)
```

```
## 'data.frame':    1470 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
##  $ 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 4 5 7 8 10 11 12 13 ...
##  $ 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 ...
```

```r
# change type chr to factor
attribution$Attrition = as.factor(attribution$Attrition)
attribution$BusinessTravel = as.factor(attribution$BusinessTravel)
attribution$Department = as.factor(attribution$Department)
attribution$EducationField = as.factor(attribution$EducationField)
attribution$Gender = as.factor(attribution$Gender)
attribution$JobRole = as.factor(attribution$JobRole)
attribution$MaritalStatus = as.factor(attribution$MaritalStatus)
attribution$OverTime = as.factor(attribution$OverTime)
```

```r
# check na
any(is.na(attribution))
```

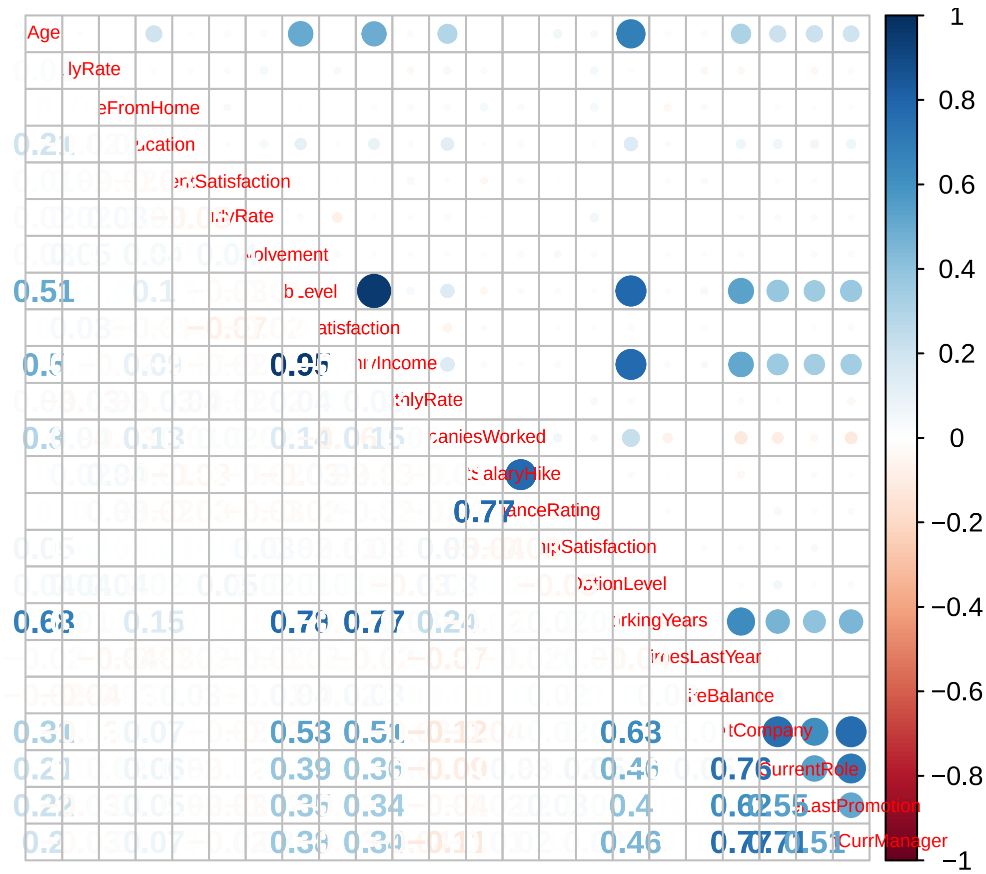
```
## [1] FALSE
```

```r
#remove Over18, EmployeeCount, EmployeeNumber, standard hours
attribution = attribution[ , c(-9,-10,-22, -27)]
```

Visualization

check correlation among numerical variables As a result: JobLevel & MonthlyIncome: 0.95 JobLevel & WorkingYears: 0.78 PercentSalaryHike & Performance Rating: 0.77 MonthlyIncome & Working Years: 0.77 YearsatCompany & YearsCurrManager: 0.77 YearsatCompany & YearInCurrentRole: 0.76 YearInCurrentRole & YearsCurrManager: 0.71 Age & WorkingYears: 0.68 WorkingYears & YearsatCompany: 0.63 YearsatCompany & YearssinceLastPromotion: 0.62

```r
library(corrplot)
numerical = unlist(lapply(attribution, is.numeric))
M = cor(attribution[, numerical])
corrplot.mixed(M, tl.cex=0.6)
```



Model

We can consider logistic regression, LDA, boosting, tree, random forest, knn.

Note: qda model is not used because of rank deficiency.

For each model, we tested different threshold and pick the one that makes the best prediction.

We use accuracy, percentage of people who leaves / people we predict to leave, and percentage of people leave / people we predict to stay to compare the models because our emphasis on predicting people who intend to leave the company.

```
calc_acc = function(actual, predicted) {
  mean(actual == predicted)
}
# how many people actually leaves when we predicted them to leave
calc_stay = function(TP, FP){
  TP / (TP + FP)
}
calc_leave = function(y, x){
  x / (x + y)
}
```

We use 70% of our dataset as traning data.

```
set.seed(432)
index = sample(nrow(attribution), size = trunc(0.7 * nrow(attribution)))
train_data= attribution[index, ]
test_data = attribution[-index, ]
```

Logistic

First, We start with Logistic Regression and we used mixed selection to get 18 important variables. From the model summary, we can see that each variable is significant.

```
set.seed(432)
null = glm(Attrition~1, data=train_data, family="binomial")
log.fit =glm(Attrition~., data=train_data, family="binomial")
regboth = step(null, scope=formula(log.fit), direction="both", trace=0)
log.fit = glm(Attrition ~ OverTime + JobRole + JobInvolvement + MaritalStatus +
                JobSatisfaction + EnvironmentSatisfaction +
                BusinessTravel + DistanceFromHome + YearsInCurrentRole +
                YearsSinceLastPromotion + TrainingTimesLastYear + Age +
                NumCompaniesWorked + RelationshipSatisfaction + WorkLifeBalance +
                YearsWithCurrManager + YearsAtCompany + TotalWorkingYears,
              data=train_data, family = "binomial")
summary(log.fit)
```

```
##
## Call:
## glm(formula = Attrition ~ OverTime + JobRole + JobInvolvement +
##     MaritalStatus + JobSatisfaction + EnvironmentSatisfaction +
##     BusinessTravel + DistanceFromHome + YearsInCurrentRole +
##     YearsSinceLastPromotion + TrainingTimesLastYear + Age + NumCompaniesWorked +
##     RelationshipSatisfaction + WorkLifeBalance + YearsWithCurrManager +
##     YearsAtCompany + TotalWorkingYears, family = "binomial",
##     data = train_data)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8729  -0.4979  -0.2467  -0.0899   3.2222
##
## Coefficients:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       0.99979    1.12183   0.891 0.372817
## OverTimeYes                       2.10223    0.23751   8.851  < 2e-16 ***
## JobRoleHuman Resources            1.57058    0.66705   2.355 0.018546 *
## JobRoleLaboratory Technician      1.41633    0.49442   2.865 0.004175 **
## JobRoleManager                    0.48063    0.75034   0.641 0.521816
## JobRoleManufacturing Director    -0.16765    0.63244  -0.265 0.790938
## JobRoleResearch Director         -1.59394    1.21943  -1.307 0.191172
## JobRoleResearch Scientist         0.28258    0.49757   0.568 0.570093
## JobRoleSales Executive            1.11301    0.47592   2.339 0.019353 *
## JobRoleSales Representative       1.92596    0.57585   3.345 0.000824 ***
## JobInvolvement                   -0.61042    0.14931  -4.088 4.35e-05 ***
## MaritalStatusMarried              0.47576    0.31400   1.515 0.129737
## MaritalStatusSingle               1.53499    0.32505   4.722 2.33e-06 ***
## JobSatisfaction                  -0.35508    0.09710  -3.657 0.000255 ***
## EnvironmentSatisfaction          -0.39092    0.09802  -3.988 6.66e-05 ***
## BusinessTravelTravel_Frequently   2.00391    0.54463   3.679 0.000234 ***
## BusinessTravelTravel_Rarely       1.19438    0.50598   2.361 0.018249 *
## DistanceFromHome                  0.05322    0.01290   4.126 3.69e-05 ***
## YearsInCurrentRole               -0.13467    0.05571  -2.417 0.015642 *
## YearsSinceLastPromotion           0.15101    0.05021   3.007 0.002636 **
## TrainingTimesLastYear            -0.18098    0.08631  -2.097 0.036006 *
## Age                              -0.02984    0.01589  -1.878 0.060378 .
## NumCompaniesWorked                0.18635    0.04520   4.123 3.74e-05 ***
## RelationshipSatisfaction         -0.31144    0.09807  -3.176 0.001494 **
## WorkLifeBalance                  -0.24699    0.15000  -1.647 0.099639 .
## YearsWithCurrManager             -0.12106    0.05813  -2.083 0.037284 *
## YearsAtCompany                    0.06879    0.04816   1.428 0.153218
## TotalWorkingYears                -0.06762    0.03285  -2.059 0.039540 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 896.03  on 1028  degrees of freedom
## Residual deviance: 594.20  on 1001  degrees of freedom
## AIC: 650.2
##
## Number of Fisher Scoring iterations: 7
```

```r
log_pred = ifelse(predict(log.fit, newdata = test_data, type = "response") >= 0.7, 'Yes','No')
t1 = table(predicted = log_pred, actual = test_data$Attrition)
t1
```

```
##          actual
## predicted  No Yes
##       No  364  55
##       Yes   2  20
```

```
log_accu = calc_acc(actual = test_data$Attrition, predicted = log_pred)
log_stay = calc_stay(t1[2,2], t1[2,1])
log_leave = calc_leave(t1[1,1], t1[1,2])
```

Linear Discriminant Analysis

Then we tried lda and this is how we did varaible selection. We took out variables that have a high correlation which can be found in visualization. Then we tried to take out variables that has little or no relationship with Attrition.

```
set.seed(432)
lda_0 = lda(Attrition~., data = train_data, prior=c(864, 166)/1030)
lda_pred0 = ifelse(predict(lda_0, test_data)$posterior[ ,1]>=0.35, 'No', 'Yes')
calc_acc(actual = test_data$Attrition,
         predicted = lda_pred0)
```

```
## [1] 0.8707483
```

```
table(predicted = lda_pred0, actual = test_data$Attrition)
```

```
##          actual
## predicted  No Yes
##       No  361  52
##       Yes   5  23
```

```
set.seed(432)
lda_1 = lda(Attrition~.- MonthlyIncome, data = train_data, prior=c(864, 166)/1030)
lda_pred1 = ifelse(predict(lda_1, test_data)$posterior[ ,1]>=0.35, 'No', 'Yes')
calc_acc(actual = test_data$Attrition,
         predicted = lda_pred1)
```

```
## [1] 0.8707483
```

```
table(predicted = lda_pred1, actual = test_data$Attrition)
```

```
##          actual
## predicted  No Yes
##       No  361  52
##       Yes   5  23
```

```
set.seed(432)
lda_2 = lda(Attrition~.- MonthlyIncome - PerformanceRating, data = train_data, prior=c(864,166)/1030)
lda_pred2 = ifelse(predict(lda_2, test_data)$posterior[ ,1]>=0.35, 'No', 'Yes')
calc_acc(actual = test_data$Attrition,
         predicted = lda_pred2)
```

```
## [1] 0.8707483
```

```r
table(predicted = lda_pred2, actual = test_data$Attrition)
```

```
##          actual
## predicted  No Yes
##       No  361  52
##       Yes   5  23
```

```r
set.seed(432)
lda_3 = lda(Attrition~.- MonthlyIncome - PerformanceRating - Education - Department - Gender,data = trai
lda_pred3 = ifelse(predict(lda_3, test_data)$posterior[ ,1]>=0.35, 'No', 'Yes')
t2=table(predicted = lda_pred3, actual = test_data$Attrition)
t2
```

```
##          actual
## predicted  No Yes
##       No  361  52
##       Yes   5  23
```

```r
lda_accu = calc_acc(actual = test_data$Attrition, predicted = lda_pred3)
lda_stay = calc_stay(t2[2,2], t2[2,1])
lda_leave = calc_leave(t2[1,1], t2[1,2])
```

Decision Tree

```r
set.seed(432)
tree_1 = tree(Attrition~., data = train_data)
summary(tree_1)
```

```
##
## Classification tree:
## tree(formula = Attrition ~ ., data = train_data)
## Variables actually used in tree construction:
##  [1] "OverTime"                 "TotalWorkingYears"
##  [3] "HourlyRate"               "JobRole"
##  [5] "StockOptionLevel"         "NumCompaniesWorked"
##  [7] "Age"                      "DistanceFromHome"
##  [9] "RelationshipSatisfaction" "MonthlyIncome"
## [11] "YearsAtCompany"
## Number of terminal nodes:  24
## Residual mean deviance:  0.5095 = 512.1 / 1005
## Misclassification error rate: 0.1079 = 111 / 1029
```

```r
tree_pred = ifelse(predict(tree_1, test_data, type="vector")[ ,1] >= 0.65, 'No', 'Yes')
t3 = table(tree_pred, test_data$Attrition)
t3
```

```
##
## tree_pred  No Yes
##       No  315  39
##       Yes  51  36
```

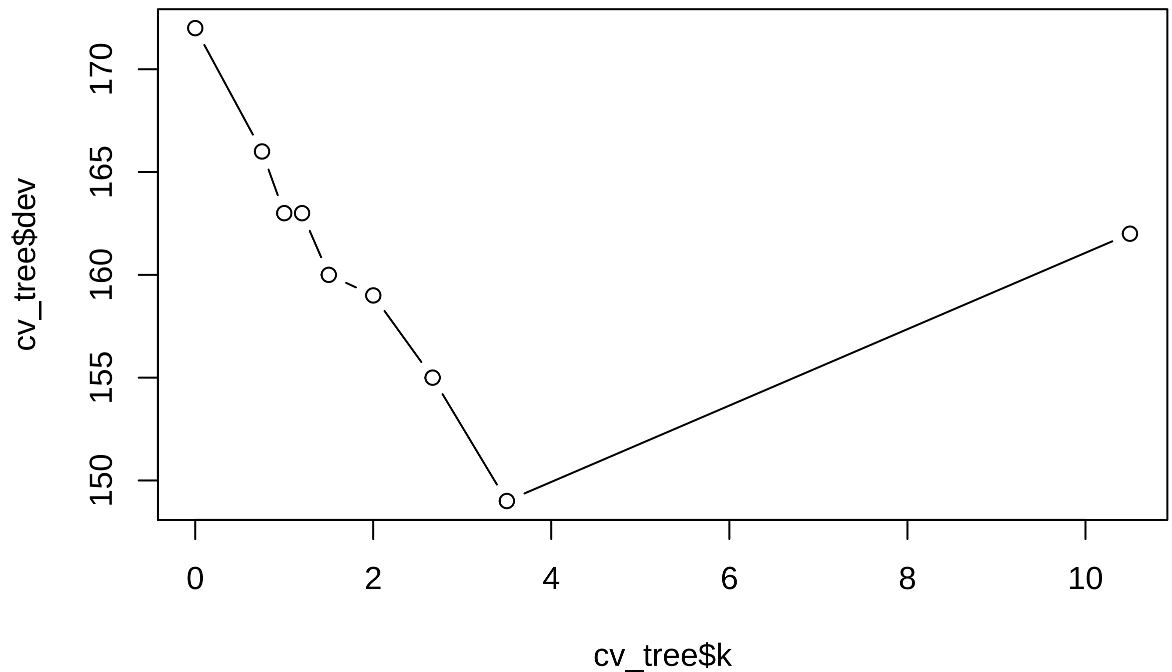
Here we tried to prune the decision tree and we pick best=3 according to the following.

```
set.seed(432)
cv_tree = cv.tree(tree_1, FUN=prune.misclass)
cv_tree
```

```
## $size
##  [1] 24 21 17 16 11  9  8  5  3  1
##
## $dev
##  [1] 172 172 166 163 163 160 159 155 149 162
##
## $k
##  [1]      -Inf  0.000000  0.750000  1.000000  1.200000  1.500000  2.000000
##  [8]  2.666667  3.500000 10.500000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cv_tree$size, cv_tree$dev , type ="b")
```

```r
plot(cv_tree$k, cv_tree$dev , type ="b")
```



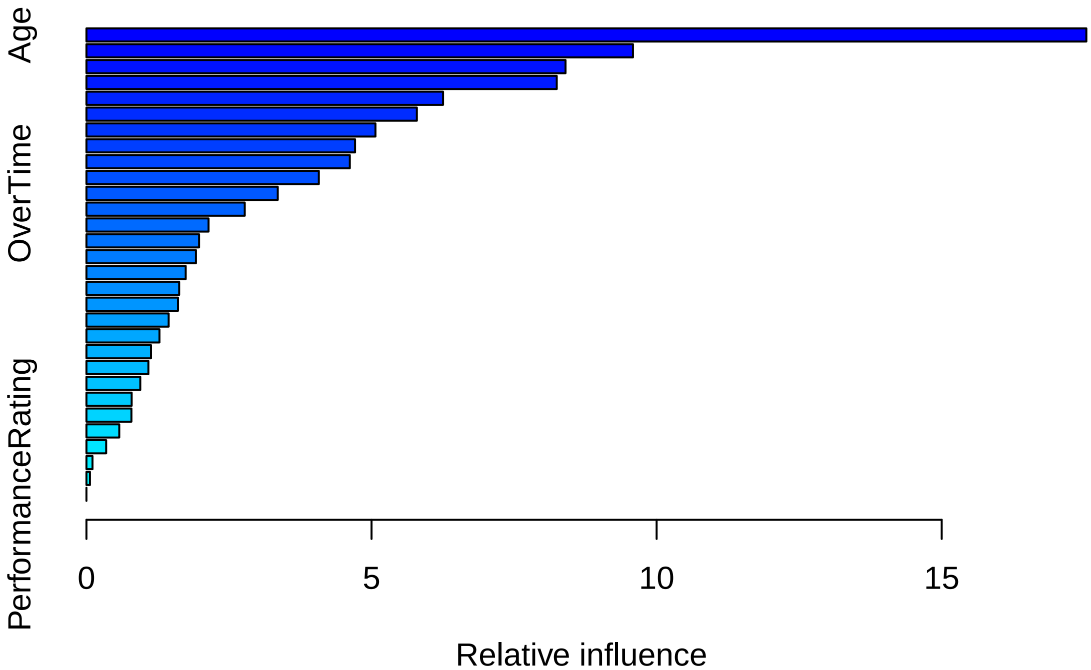

```r
set.seed(432)
prune_tree = prune.misclass (tree_1 , best = 3)
tree_pred = ifelse(predict(prune_tree, test_data, type="vector")[ ,1] >= 0.5, 'No', 'Yes')
table(tree_pred, test_data$Attrition)
```

```
##
## tree_pred  No Yes
##       No  357  61
##       Yes   9  14
```

The pruned tree is not good as the original one.

Boosting

```r
set.seed(432)
train_data_copy = train_data
train_data_copy$Attrition = ifelse(train_data_copy$Attrition == "No", 0, 1)
boosting_1 = gbm(Attrition~., data=train_data_copy, distribution="bernoulli", n.trees=1000, shrinkage =
summary(boosting_1)
```




```
##                                                 var     rel.inf
## Age                                             Age 17.53694587
## MonthlyIncome                         MonthlyIncome  9.58510244
## DailyRate                                 DailyRate  8.40256253
## JobRole                                     JobRole  8.24849537
## MonthlyRate                             MonthlyRate  6.25456386
## EducationField                       EducationField  5.79492256
## TotalWorkingYears                 TotalWorkingYears  5.06889789
## YearsAtCompany                       YearsAtCompany  4.71110013
## HourlyRate                               HourlyRate  4.61943287
## DistanceFromHome                   DistanceFromHome  4.07584064
## OverTime                                   OverTime  3.35547551
## PercentSalaryHike                 PercentSalaryHike  2.77684576
## RelationshipSatisfaction RelationshipSatisfaction  2.14165950
## YearsSinceLastPromotion   YearsSinceLastPromotion  1.97548383
## TrainingTimesLastYear       TrainingTimesLastYear  1.92084446
## NumCompaniesWorked             NumCompaniesWorked  1.74115141
## EnvironmentSatisfaction   EnvironmentSatisfaction  1.62619840
## YearsWithCurrManager         YearsWithCurrManager  1.60587116
## JobInvolvement                       JobInvolvement  1.44246482
## StockOptionLevel                   StockOptionLevel  1.28053282
## Education                                 Education  1.13221267
## WorkLifeBalance                     WorkLifeBalance  1.08663828
## JobSatisfaction                     JobSatisfaction  0.94434207
## YearsInCurrentRole               YearsInCurrentRole  0.79339741
## BusinessTravel                       BusinessTravel  0.78914972
```

```
## JobLevel                                     JobLevel  0.57591325
## MaritalStatus                           MaritalStatus  0.34596604
## Department                                 Department  0.10613832
## Gender                                         Gender  0.06185038
## PerformanceRating                   PerformanceRating  0.00000000
```

```
boo_pred = ifelse(predict(boosting_1, newdata = test_data, n.trees = 1000, type="response")>0.45, 'Yes'
table(predicted = boo_pred, actual = test_data$Attrition)
```

```
##          actual
## predicted  No Yes
##       No  341  37
##       Yes  25  38
```

After we adjust the shrinkage and n.minobsinnode.

```
set.seed(432)
boosting_1 = gbm(Attrition~.-PerformanceRating-Gender-Department, data=train_data_copy, distribution =
boo_pred = ifelse(predict(boosting_1, newdata = test_data, n.trees = 1000, type="response")>0.45, 'Yes'
t6=table(predicted = boo_pred, actual = test_data$Attrition)
t6
```

```
##          actual
## predicted  No Yes
##       No  346  38
##       Yes  20  37
```

Random Forest

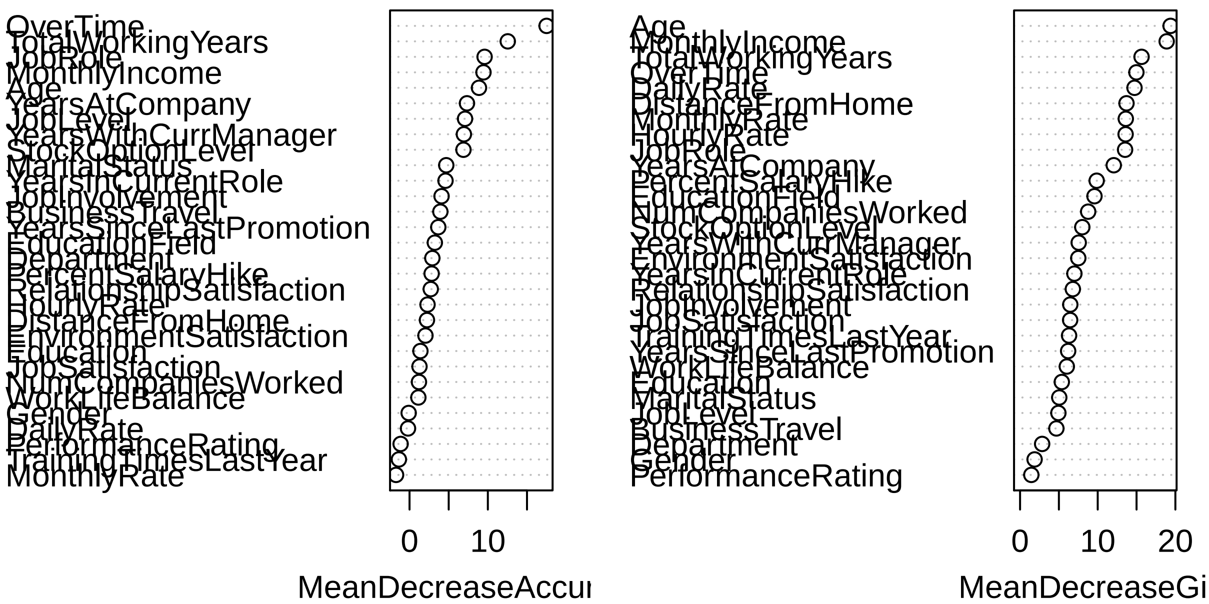
We start by fitting all the variables in the model and then take out the least important variable one by one.

```
set.seed(432)
rf_1 = randomForest(Attrition~., data = train_data, importance=TRUE)
rf_pred = ifelse(predict(rf_1, newdata = test_data, type = "prob")[ ,1] >= 0.7, 'No', 'Yes')
table(rf_pred, test_data$Attrition)
```

```
##
## rf_pred  No Yes
##     No  342  42
##     Yes  24  33
```

```
varImpPlot(rf_1)
```

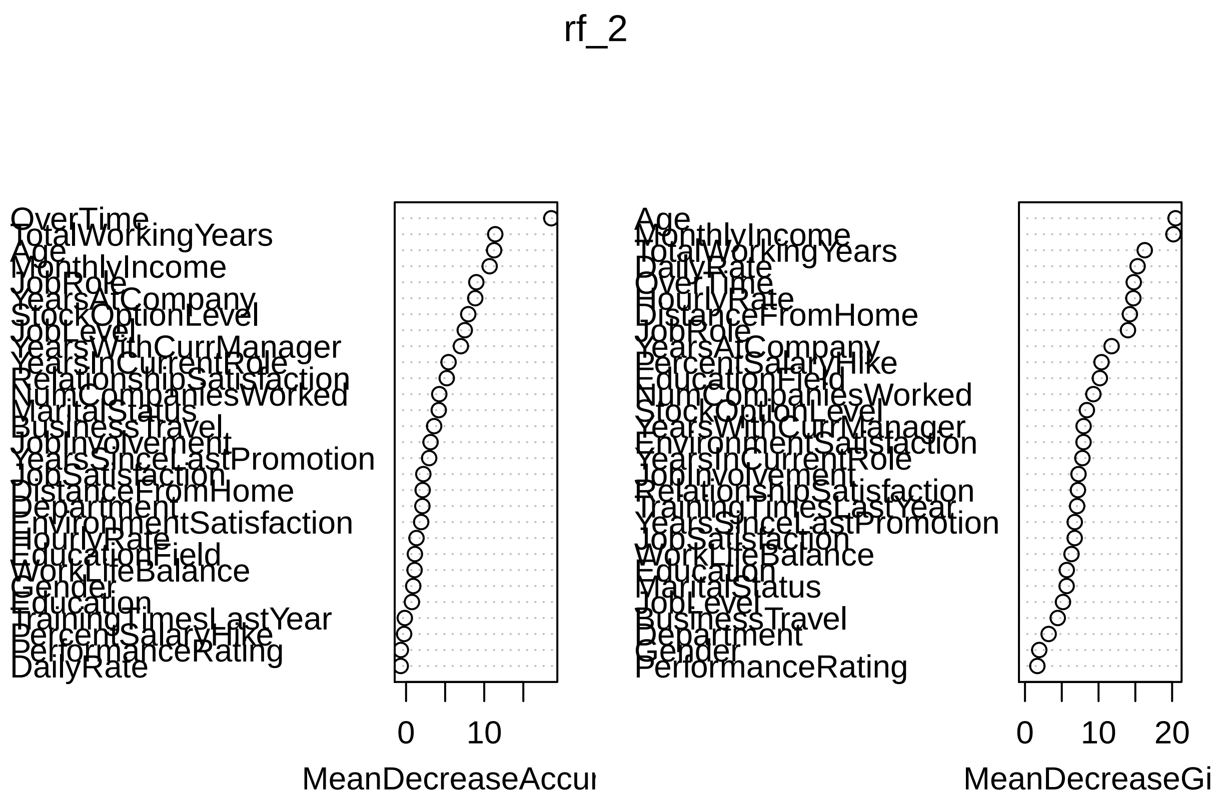
rf_1




```r
set.seed(432)
rf_2 = randomForest(Attrition~.-MonthlyRate, data = train_data, importance=TRUE)
rf_pred = ifelse(predict(rf_2, newdata = test_data, type = "prob")[ ,1] >= 0.7, 'No', 'Yes')
table(rf_pred, test_data$Attrition)
```

```
##
## rf_pred  No Yes
##     No  341  42
##     Yes  25  33
```

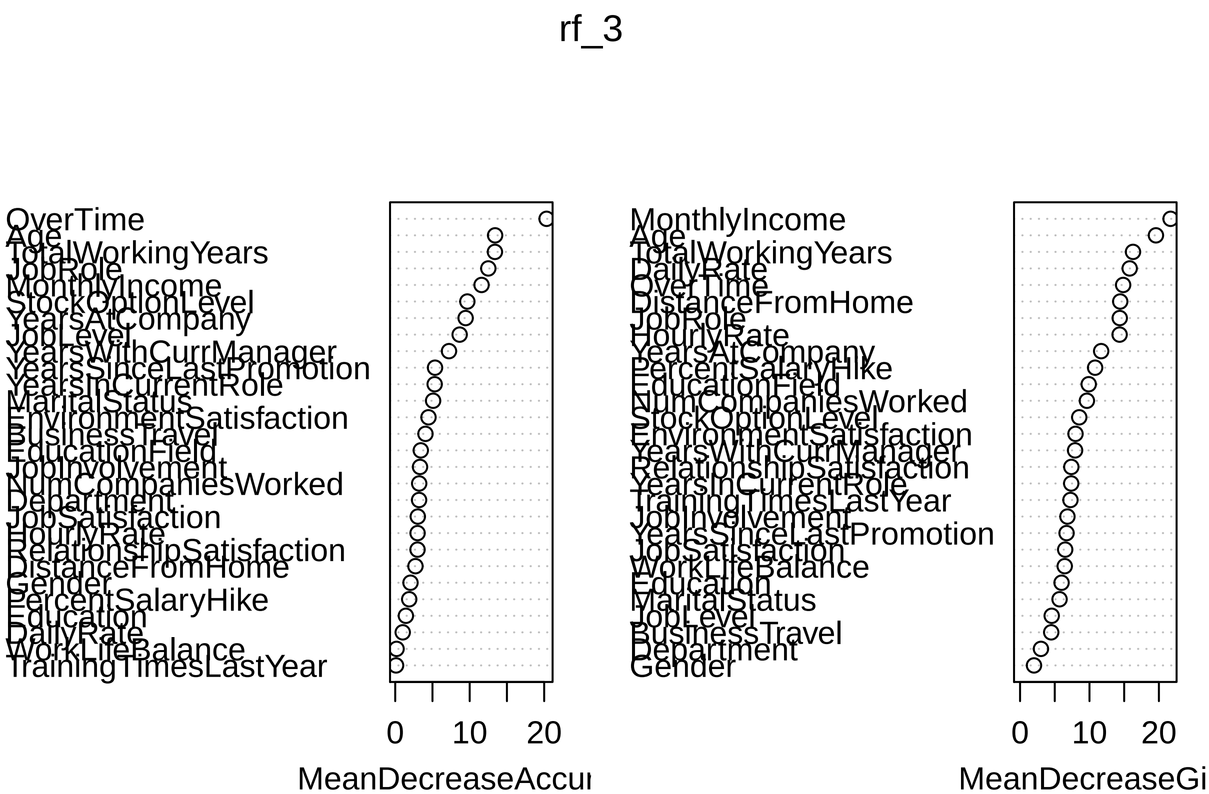
```r
varImpPlot(rf_2)
```




```r
set.seed(432)
rf_3 = randomForest(Attrition~.-MonthlyRate - PerformanceRating, data = train_data, importance = TRUE)
rf_pred = ifelse(predict(rf_3, newdata = test_data, type = "prob")[ ,1] >= 0.7, 'No', 'Yes')
table(rf_pred, test_data$Attrition)
```

```
##
## rf_pred  No Yes
##     No  336  41
##     Yes  30  34
```

```r
varImpPlot(rf_3)
```



KNN

We first put all the predictors into the knn model.

```
set.seed(56)
knnFit <- train(Attrition ~ ., data = train_data, method = "knn", trControl = trainControl(method = "cv
table(predicted = ifelse(predict(knnFit, test_data, type = "prob")[ ,1]>=0.7, 'No', 'Yes'),
      actual = test_data$Attrition)
```

```
##          actual
## predicted  No Yes
##       No  352  58
##       Yes  14  17
```

```
set.seed(56)
knnFit1 <- train(Attrition ~ .- JobLevel - Gender - DailyRate - Department - PercentSalaryHike - NumComp
knn_pred = ifelse(predict(knnFit1, test_data, type = "prob")[ ,1]>=0.7, 'No', 'Yes')
t5 = table(predicted = ifelse(predict(knnFit1, test_data, type = "prob")[ ,1]>=0.6, 'No','Yes'),
           actual = test_data$Attrition)
t5
```

```
##          actual
## predicted  No Yes
##       No  364  64
##       Yes   2  11
```

"The strength of naive Bayes comes from its ability to handle a large number of predictors, p, even with a limited sample size n. Even with the naive independence assumption, naive Bayes works rather well in practice. Also because of this assumption, we can often train naive Bayes where LDA and QDA may be impossible to train because of the large number of parameters relative to the number of observations."

we give naivebayes a try here. We fit a full model first and then we fit a very simple model with only 5 predictors.

naivebayes

```
set.seed(56)
nb_1 = naiveBayes(Attrition~., data = train_data, prior=c(864, 166)/1030)
nb_pred = ifelse(predict(nb_1, test_data, type = "raw")[ ,1]>=0.1, 'No', 'Yes')
calc_acc(nb_pred, test_data$Attrition)
```

```
## [1] 0.8503401
```

```
t7=table(predicted = nb_pred, actual = test_data$Attrition)
t7
```

```
##          actual
## predicted  No Yes
##       No  356  56
##       Yes  10  19
```

```
nb_accu = calc_acc(actual = test_data$Attrition, predicted = nb_pred)
nb_stay = calc_stay(t7[2,2], t7[2,1])
nb_leave = calc_leave(t7[1,1], t7[1,2])
```

Model Comparison

Model Name	Model Formula	Accuracy	Percentage of people leave if we predict they will leave	Percentage of People leave if we predict they will stay
Logistic Regression	'Attrition~.-DailyRate- Department- Education- EducationField- Gender- HourlyRate- JobLevel- MonthlyIncome- MonthlyRate- PercentSalaryHike- PerformanceRating- StockOptionLevel'	0.871	0.909	0.131
Linear Discriminent Analysis	'Attrition~.-MonthlyIncome- PerformanceRating- Education- Department- Gender'	0.871	0.821	0.126
Tree	'Attrition ~ .'	0.841	0.414	0.110
Random Forest	'Attrition~.-MonthlyRate- PerformanceRating- TrainingTimes- LastYear- Gender- WorkLifeBalance- PercentSalaryHike- DailyRate'	0.871	0.679	0.136
knn	'Attrition~.-JobLevel- Gender- DailyRate- Department- Per- centSalaryHike'	0.844	0.667	0.141

Model Name	Model Formula	Accuracy	Percentage of people leave if we predict they will leave	Percentage of People leave if we predict they will stay
Boosting	‘Attrition~.-PerformanceRating- Gender-Department’	0.859	0.760	0.135
Naive Bayes	‘Attrition~.’	0.850	0.655	0.136