Credit Random Forest

Lei Deng, ti7597, STAT6620

May 18, 2017

## Step 1 - Colleting data

Read the data to the R

# load the necessary library  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(randomForest)

## 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(Amelia)

## Loading required package: Rcpp

## ##   
## ## Amelia II: Multiple Imputation  
## ## (Version 1.7.4, built: 2015-12-05)  
## ## Copyright (C) 2005-2017 James Honaker, Gary King and Matthew Blackwell  
## ## Refer to http://gking.harvard.edu/amelia/ for more information  
## ##

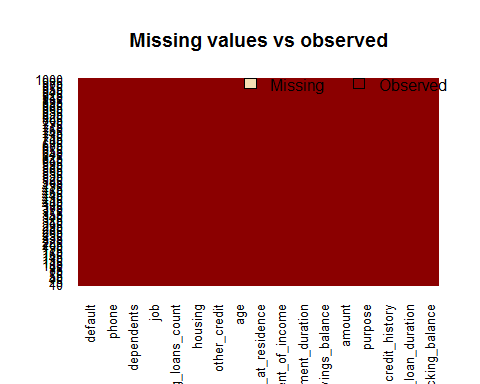
credit <- read.csv("credit.csv")

## Step 2 - Exploring and preparing the data

str(credit)

## 'data.frame': 1000 obs. of 17 variables:  
## $ checking\_balance : Factor w/ 4 levels "< 0 DM","> 200 DM",..: 1 3 4 1 1 4 4 3 4 3 ...  
## $ months\_loan\_duration: int 6 48 12 42 24 36 24 36 12 30 ...  
## $ credit\_history : Factor w/ 5 levels "critical","good",..: 1 2 1 2 4 2 2 2 2 1 ...  
## $ purpose : Factor w/ 6 levels "business","car",..: 5 5 4 5 2 4 5 2 5 2 ...  
## $ amount : int 1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...  
## $ savings\_balance : Factor w/ 5 levels "< 100 DM","> 1000 DM",..: 5 1 1 1 1 5 4 1 2 1 ...  
## $ employment\_duration : Factor w/ 5 levels "< 1 year","> 7 years",..: 2 3 4 4 3 3 2 3 4 5 ...  
## $ percent\_of\_income : int 4 2 2 2 3 2 3 2 2 4 ...  
## $ years\_at\_residence : int 4 2 3 4 4 4 4 2 4 2 ...  
## $ age : int 67 22 49 45 53 35 53 35 61 28 ...  
## $ other\_credit : Factor w/ 3 levels "bank","none",..: 2 2 2 2 2 2 2 2 2 2 ...  
## $ housing : Factor w/ 3 levels "other","own",..: 2 2 2 1 1 1 2 3 2 2 ...  
## $ existing\_loans\_count: int 2 1 1 1 2 1 1 1 1 2 ...  
## $ job : Factor w/ 4 levels "management","skilled",..: 2 2 4 2 2 4 2 1 4 1 ...  
## $ dependents : int 1 1 2 2 2 2 1 1 1 1 ...  
## $ phone : Factor w/ 2 levels "no","yes": 2 1 1 1 1 2 1 2 1 1 ...  
## $ default : Factor w/ 2 levels "no","yes": 1 2 1 1 2 1 1 1 1 2 ...

# Check if there are any missing values  
  
missmap(credit, main = "Missing values vs observed")



# number of missing values in each column  
  
sapply(credit,function(x) sum(is.na(x)))

## checking\_balance months\_loan\_duration credit\_history   
## 0 0 0   
## purpose amount savings\_balance   
## 0 0 0   
## employment\_duration percent\_of\_income years\_at\_residence   
## 0 0 0   
## age other\_credit housing   
## 0 0 0   
## existing\_loans\_count job dependents   
## 0 0 0   
## phone default   
## 0 0

# number of unique values in each column  
  
sapply(credit, function(x) length(unique(x)))

## checking\_balance months\_loan\_duration credit\_history   
## 4 33 5   
## purpose amount savings\_balance   
## 6 921 5   
## employment\_duration percent\_of\_income years\_at\_residence   
## 5 4 4   
## age other\_credit housing   
## 53 3 3   
## existing\_loans\_count job dependents   
## 4 4 2   
## phone default   
## 2 2

## Step3 - Training a model on the data

set.seed(300)  
rf <- randomForest(default ~ ., data = credit)  
rf

##   
## Call:  
## randomForest(formula = default ~ ., data = credit)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 4  
##   
## OOB estimate of error rate: 23.8%  
## Confusion matrix:  
## no yes class.error  
## no 640 60 0.08571429  
## yes 178 122 0.59333333

## step4 - Evaluating model performance

ctrl <- trainControl(method = "repeatedcv",  
 number = 10, repeats = 10)  
  
# auto-tune a random forest  
grid\_rf <- expand.grid(.mtry = c(2, 4, 8, 16))  
  
set.seed(300)  
m\_rf <- train(default ~ ., data = credit, method = "rf",  
 metric = "Kappa", trControl = ctrl,  
 tuneGrid = grid\_rf)  
m\_rf

## Random Forest   
##   
## 1000 samples  
## 16 predictor  
## 2 classes: 'no', 'yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.7256 0.1311283  
## 4 0.7476 0.2878470  
## 8 0.7519 0.3346061  
## 16 0.7557 0.3618152  
##   
## Kappa was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 16.

## step5 - Improving model performance

# auto-tune a boosted C5.0 decision tree  
grid\_c50 <- expand.grid(.model = "tree",  
 .trials = c(10, 20, 30, 40),  
 .winnow = "FALSE")  
  
set.seed(300)  
m\_c50 <- train(default ~ ., data = credit, method = "C5.0",  
 metric = "Kappa", trControl = ctrl,  
 tuneGrid = grid\_c50)

## Loading required package: C50

## Loading required package: plyr

## Warning in Ops.factor(x$winnow): '!' not meaningful for factors

m\_c50

## C5.0   
##   
## 1000 samples  
## 16 predictor  
## 2 classes: 'no', 'yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...   
## Resampling results across tuning parameters:  
##   
## trials Accuracy Kappa   
## 10 0.7325 0.3215655  
## 20 0.7343 0.3268052  
## 30 0.7381 0.3343137  
## 40 0.7388 0.3335082  
##   
## Tuning parameter 'model' was held constant at a value of tree  
##   
## Tuning parameter 'winnow' was held constant at a value of FALSE  
## Kappa was used to select the optimal model using the largest value.  
## The final values used for the model were trials = 30, model = tree  
## and winnow = FALSE.