Coursera - Practical Machine Learning Project

Stefan Krähenmann

Saturday, September 07, 2019

## Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These types of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it.

## The target of the project

In this project, the goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: <http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har> (see the section on the Weight Lifting Exercise Dataset).

## Libraries

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(parallel)  
library(doParallel)

## Loading required package: foreach

## Loading required package: iterators

library(ggplot2)  
library(RANN)  
library(rattle)

## Rattle: A free graphical interface for data science with R.  
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.  
## Geben Sie 'rattle()' ein, um Ihre Daten mischen.

library(corrplot)

## corrplot 0.84 loaded

## Loading the data from URL

training <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),header=TRUE)  
testdata <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),header=TRUE)

# General overview of the dataset

dim(training)

## [1] 19622 160

str(training)

## 'data.frame': 19622 obs. of 160 variables:  
## $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ user\_name : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...  
## $ raw\_timestamp\_part\_1 : int 1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 ...  
## $ raw\_timestamp\_part\_2 : int 788290 808298 820366 120339 196328 304277 368296 440390 484323 484434 ...  
## $ cvtd\_timestamp : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...  
## $ new\_window : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ num\_window : int 11 11 11 12 12 12 12 12 12 12 ...  
## $ roll\_belt : num 1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...  
## $ pitch\_belt : num 8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...  
## $ yaw\_belt : num -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...  
## $ total\_accel\_belt : int 3 3 3 3 3 3 3 3 3 3 ...  
## $ kurtosis\_roll\_belt : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ kurtosis\_picth\_belt : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ kurtosis\_yaw\_belt : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_roll\_belt : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_roll\_belt.1 : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_yaw\_belt : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...  
## $ max\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ max\_picth\_belt : int NA NA NA NA NA NA NA NA NA NA ...  
## $ max\_yaw\_belt : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ min\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_pitch\_belt : int NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_yaw\_belt : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ amplitude\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ amplitude\_pitch\_belt : int NA NA NA NA NA NA NA NA NA NA ...  
## $ amplitude\_yaw\_belt : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ var\_total\_accel\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ avg\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ stddev\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ var\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ avg\_pitch\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ stddev\_pitch\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ var\_pitch\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ avg\_yaw\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ stddev\_yaw\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ var\_yaw\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ gyros\_belt\_x : num 0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...  
## $ gyros\_belt\_y : num 0 0 0 0 0.02 0 0 0 0 0 ...  
## $ gyros\_belt\_z : num -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...  
## $ accel\_belt\_x : int -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...  
## $ accel\_belt\_y : int 4 4 5 3 2 4 3 4 2 4 ...  
## $ accel\_belt\_z : int 22 22 23 21 24 21 21 21 24 22 ...  
## $ magnet\_belt\_x : int -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...  
## $ magnet\_belt\_y : int 599 608 600 604 600 603 599 603 602 609 ...  
## $ magnet\_belt\_z : int -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...  
## $ roll\_arm : num -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...  
## $ pitch\_arm : num 22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...  
## $ yaw\_arm : num -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...  
## $ total\_accel\_arm : int 34 34 34 34 34 34 34 34 34 34 ...  
## $ var\_accel\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ avg\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ stddev\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ var\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ avg\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ stddev\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ var\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ avg\_yaw\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ stddev\_yaw\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ var\_yaw\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ gyros\_arm\_x : num 0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...  
## $ gyros\_arm\_y : num 0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...  
## $ gyros\_arm\_z : num -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...  
## $ accel\_arm\_x : int -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...  
## $ accel\_arm\_y : int 109 110 110 111 111 111 111 111 109 110 ...  
## $ accel\_arm\_z : int -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...  
## $ magnet\_arm\_x : int -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...  
## $ magnet\_arm\_y : int 337 337 344 344 337 342 336 338 341 334 ...  
## $ magnet\_arm\_z : int 516 513 513 512 506 513 509 510 518 516 ...  
## $ kurtosis\_roll\_arm : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ kurtosis\_picth\_arm : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ kurtosis\_yaw\_arm : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_roll\_arm : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_pitch\_arm : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_yaw\_arm : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ max\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ max\_picth\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ max\_yaw\_arm : int NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_yaw\_arm : int NA NA NA NA NA NA NA NA NA NA ...  
## $ amplitude\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ amplitude\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ amplitude\_yaw\_arm : int NA NA NA NA NA NA NA NA NA NA ...  
## $ roll\_dumbbell : num 13.1 13.1 12.9 13.4 13.4 ...  
## $ pitch\_dumbbell : num -70.5 -70.6 -70.3 -70.4 -70.4 ...  
## $ yaw\_dumbbell : num -84.9 -84.7 -85.1 -84.9 -84.9 ...  
## $ kurtosis\_roll\_dumbbell : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ kurtosis\_picth\_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ kurtosis\_yaw\_dumbbell : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_roll\_dumbbell : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_pitch\_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ skewness\_yaw\_dumbbell : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...  
## $ max\_roll\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...  
## $ max\_picth\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...  
## $ max\_yaw\_dumbbell : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ min\_roll\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_pitch\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_yaw\_dumbbell : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ amplitude\_roll\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...  
## [list output truncated]

## Data preparation

The training dataset comprises of 19622 observations on 160 columns. Many columns have NA values or blank values on almost every observation.

Machine learning algorithms require complete datasets missing values have to be filled by estimating their values from the remaining data. However, this may introduce some erroneous values, particularly in case of a high amount of missing values. Also predictors containing low As a consequence, predictors containing a high fraction of missing values generally not add much useful information. Also predictors with low variability do not provide much insight. Therefore, columns with many missing values (more than 95 percent missing values) and with low variance will be discared.

ind.na <- which(apply(is.na(training),c(2),mean) > .95) # check for missing data  
length(ind.na) # number of predictors with high fraction of missing values

## [1] 67

ind.nzv <- nearZeroVar(training) # check for data variance  
length(ind.nzv) # number of predictors with low variability

## [1] 60

The first seven columns give information about the participants of the test and timestamps. They will also not be taken into account.

# Combine the indexes of columns having at least 95% of missing values, low varoance and the first 7 columns  
ind.remove <- unique(c(ind.na, ind.nzv, 1:7))  
training <- training[,-ind.remove]  
name <- names(training)

# Partition the training dataset into new training and validtion dataset

set.seed(1201)  
inTrain <- createDataPartition(training$classe, p=.7, list=F)  
trainingPart <- training[inTrain,]  
validationPart <- training[-inTrain,]  
dim(trainingPart)

## [1] 13737 53

dim(validationPart)

## [1] 5885 53

This allows for an in sample validation of the machine learning algorithms. The new training dataset amounts to 70 percent of the original training dataset.

## Model buidling

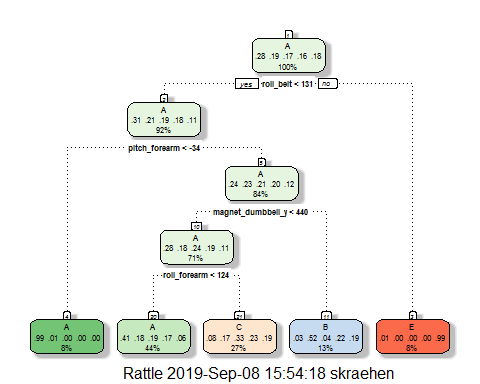
For this project I used three different models, classification trees, random forests and generalized boosted models for prediction. 1.classification trees 2.random forests 3.Generalized Boosted Model

Cross-validation is used with 5 folds to limit the effects of overfitting and to improve the efficicency of the models. As relatievly complex machine learning algorithms are tested on a large dataset with many predictors, the processing time can be large. Therefor, calculations are done parallelly. All this is handelded by the parameter “trainingControl” which is then passed to caret’s “train” function.

myControl <- trainControl(method = "cv",  
 number = 5,  
 allowParallel = TRUE,  
 verboseIter = FALSE)  
# Allow for parallel calculations  
cluster <- makeCluster(detectCores() - 3) # leave 3 core for OS  
registerDoParallel(cluster)

## Prediction with classification trees

set.seed(12)  
model\_tree <- train(classe~., trainingPart, method="rpart", trControl = myControl, tuneLength=3)  
fancyRpartPlot(model\_tree$finalModel)



trainpred.tree <- predict(model\_tree,newdata=trainingPart)  
confMat.tree <- confusionMatrix(trainingPart$classe,trainpred.tree)  
# display confusion matrix and model accuracy  
confMat.tree$table

## Reference  
## Prediction A B C D E  
## A 3548 52 295 0 11  
## B 1118 906 634 0 0  
## C 1116 73 1207 0 0  
## D 996 390 866 0 0  
## E 361 333 693 0 1138

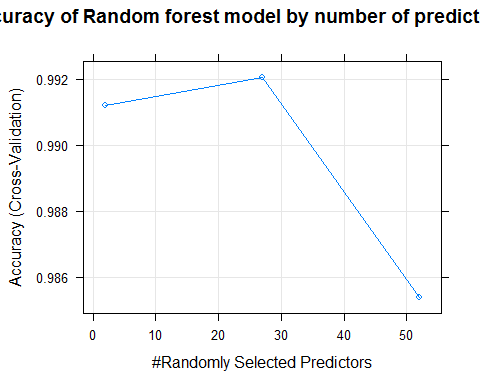
train\_acc.tree <- confMat.tree$overall[1]  
train\_acc.tree # show accuracy

## Accuracy   
## 0.4949407

The accuracy of this first model is very low (49 %). This means that the outcome class will not be predicted very well using this model.

## Prediction with random forest

set.seed(100)  
model\_rf <- train(classe~., trainingPart, method="rf", trControl = myControl, tuneLength=3, verbose=FALSE)  
plot(model\_rf,main="Accuracy of Random forest model by number of predictors")



trainpred.rf <- predict(model\_rf,newdata=trainingPart)  
confMat.rf <- confusionMatrix(trainingPart$classe,trainpred.rf)  
# display confusion matrix and model accuracy  
confMat.rf$table

## Reference  
## Prediction A B C D E  
## A 3906 0 0 0 0  
## B 0 2658 0 0 0  
## C 0 0 2396 0 0  
## D 0 0 0 2252 0  
## E 0 0 0 0 2525

train\_acc.rf <- confMat.rf$overall[1]   
train\_acc.rf # show accuracy

## Accuracy   
## 1

names(model\_rf$finalModel) # show names of chosen predictors

## [1] "call" "type" "predicted"   
## [4] "err.rate" "confusion" "votes"   
## [7] "oob.times" "classes" "importance"   
## [10] "importanceSD" "localImportance" "proximity"   
## [13] "ntree" "mtry" "forest"   
## [16] "y" "test" "inbag"   
## [19] "xNames" "problemType" "tuneValue"   
## [22] "obsLevels" "param"

# Compute the variable importance   
MostImpVars <- varImp(model\_rf)  
MostImpVars

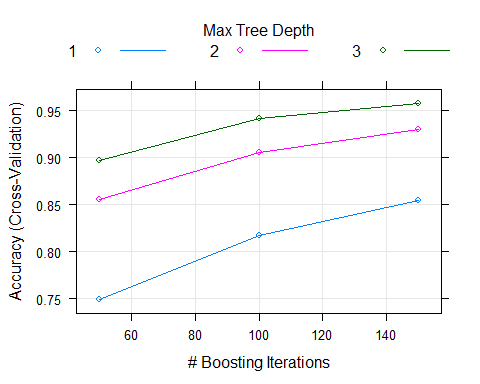
## rf variable importance  
##   
## only 20 most important variables shown (out of 52)  
##   
## Overall  
## roll\_belt 100.00  
## pitch\_forearm 61.36  
## yaw\_belt 54.96  
## magnet\_dumbbell\_y 44.13  
## magnet\_dumbbell\_z 43.14  
## pitch\_belt 42.61  
## roll\_forearm 41.74  
## accel\_dumbbell\_y 22.11  
## accel\_forearm\_x 18.32  
## roll\_dumbbell 16.83  
## magnet\_dumbbell\_x 16.11  
## magnet\_belt\_z 15.25  
## magnet\_forearm\_z 14.18  
## total\_accel\_dumbbell 13.64  
## accel\_belt\_z 13.43  
## magnet\_belt\_y 12.98  
## accel\_dumbbell\_z 12.67  
## yaw\_arm 11.27  
## magnet\_belt\_x 10.64  
## gyros\_belt\_z 10.01

Random forest reaches a much higher accuracy (100 %) using 5-fold cross-validation than with classification tree.

The optimal number of predictors (the number of predictors with the highest accuracy) is 27. There is no significal increase of the accuracy from 2 predictors to 27, but the accuracy slightly decreases with more than 27 predictors. “roll\_belt” has shown to be the most important variable, it has been chosen in each cross-validation trial.

## Prediction with gradient boosting

set.seed(35)  
model\_gbm <- train(classe~., trainingPart, method="gbm", trControl = myControl, tuneLength=3, verbose=FALSE)  
plot(model\_gbm)



trainpred.gbm <- predict(model\_gbm,newdata=trainingPart)  
confMat.gbm <- confusionMatrix(trainingPart$classe,trainpred.gbm)  
# display confusion matrix and model accuracy  
confMat.gbm$table

## Reference  
## Prediction A B C D E  
## A 3875 24 1 5 1  
## B 57 2549 49 3 0  
## C 0 59 2306 26 5  
## D 1 3 53 2183 12  
## E 3 19 15 34 2454

train\_acc.gbm <- confMat.gbm$overall[1]  
train\_acc.gbm # show accuracy

## Accuracy   
## 0.9730654

Also prediction with gradient boosting method is high (97 %).

# Predict using validation dataset  
valpred.tree <- predict(model\_tree,newdata=validationPart)  
valpred.rf <- predict(model\_rf,newdata=validationPart)  
valpred.gbm <- predict(model\_gbm,newdata=validationPart)  
  
# accuracy with classification tree  
val\_acc.tree <- confusionMatrix(validationPart$classe,valpred.tree)$overall["Accuracy"]  
val\_acc.rf <- confusionMatrix(validationPart$classe,valpred.rf)$overall["Accuracy"]  
val\_acc.gbm <- confusionMatrix(validationPart$classe,valpred.gbm)$overall["Accuracy"]

# Compare in sample error with out of sample error

df <- round(rbind(c(train\_acc.tree, train\_acc.rf, train\_acc.gbm), c(val\_acc.tree, val\_acc.rf, val\_acc.gbm)),2)  
row.names(df) <- c("in-sample", "out-of-sample")  
colnames(df) <- c("tree","rf","gbm")  
#df

|  |  |  |  |
| --- | --- | --- | --- |
|  | tree | rf | gbm |
| in-sample | 0.49 | 1.00 | 0.97 |
| out-of-sample | 0.50 | 0.99 | 0.97 |

## Conclusion

The out-of sample error is slightly higher than the in sample error. The lower in sampl error is because the models have been tuned with the training data, which always leads to some overfitting. Yet, the difference is very small here. The random forest model is overall the best one. Therefore, I will use the random forest model to predict the values of classe for the test data set.

## Application of the random forest model to the test dataset

prediction\_final <- predict(model\_rf, newdata=testdata[,-ind.remove])  
prediction\_final

## [1] B A B A A E D B A A B C B A E E A B B B  
## Levels: A B C D E

# Unregister cluster for parallel calculation  
stopCluster(cluster)  
registerDoSEQ()