Qualifying Weight Lifting Exercises

Rudy Veenhoff

25 november 2016

Introduction

This article is about qualifying if a certain weigh lifting exercise (the dumbbell curl) was executed correctly. Six participants were asked to perform one set of ten repetitions in five different ways. Data was measured via three different sensors on the body and one on the dumbbell. The aim is to use these measurements to predict which of these five executions of the exercise is being performed. This research was first carried out by Velloso, Bulling, Gellersen, Ugulino, Fuks, who presented their conclusions at ACM SIGCHI 2013. The data set for this has been generously supplied by them under the Creative Commons License and can be found here. This article will focus on reproducing the results they've presented.

Reading in the data set and preprocessing

The dataset consists of 19622 observations of 160 variables. A lot of these variables however have missing entries. Due to the sheer number of missing values in these variables, we decide to not use an imputing technique. Instead we choose to pick only the variables which have no missing observations. This results in a data set of 60 variables. We then exclude the variables for username, timestamps and windows. These are the first seven variables. This leaves us with a data set with only measurements from the sensors. The variable we wish to predict is the "classe" variable; i.e. a factor variable with 5 levels: A through E.

```
rawData <- read.csv("pml-training.csv", na.strings=c("NA",""))
dataNoNA <- rawData[,colSums(!is.na(rawData)) == nrow(rawData)]
data <- dataNoNA[,-(1:7)]
```

Splitting into training set and test set

For estimating the accuracy on the real test set we use cross validation. The data is split into a training and test set using the caret package. The accuracy on this test set will be an estimate for the accuracy on the actual test set.

```
library(caret);library(rattle);library(knitr)
set.seed(3141592)
inTrain <- createDataPartition(y=data$classe,p=3/4)[[1]]
training <- data[inTrain,]; testing <- data[-inTrain,]
```

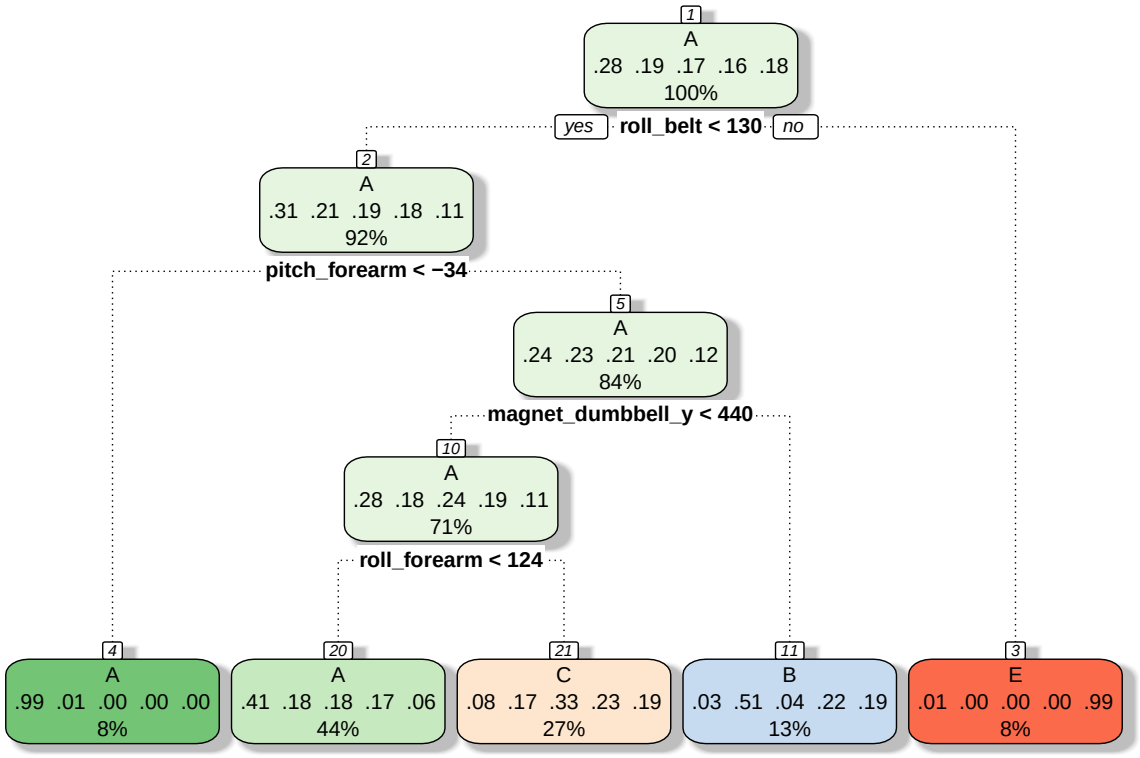
Fitting a decision tree

The first model we consider will be a decision tree.

```
set.seed(3141592)
rpartFit <- train(classe~., data=training, method="rpart")
```

The model is as follows:

```
fancyRpartPlot(rpartFit$finalModel)
```



We note that this model never predicts an outcome of classe D. This already casts a bit of doubt on the correctness of the model. Let us see how fitted model does on the testing set.

```
rpartPred <- predict(rpartFit, newdata=testing)
confMatrix <- confusionMatrix(rpartPred,testing$classe)
kable(confMatrix$table)
```

	A	B	C	D	E
A	1268	388	411	378	132
B	23	307	24	138	114
C	101	254	420	288	238
D	0	0	0	0	0
E	3	0	0	0	417

```
round(confMatrix$overall[1],2)
```

```
## Accuracy
##     0.49
```

An accuracy of 49% is not that remarkable.

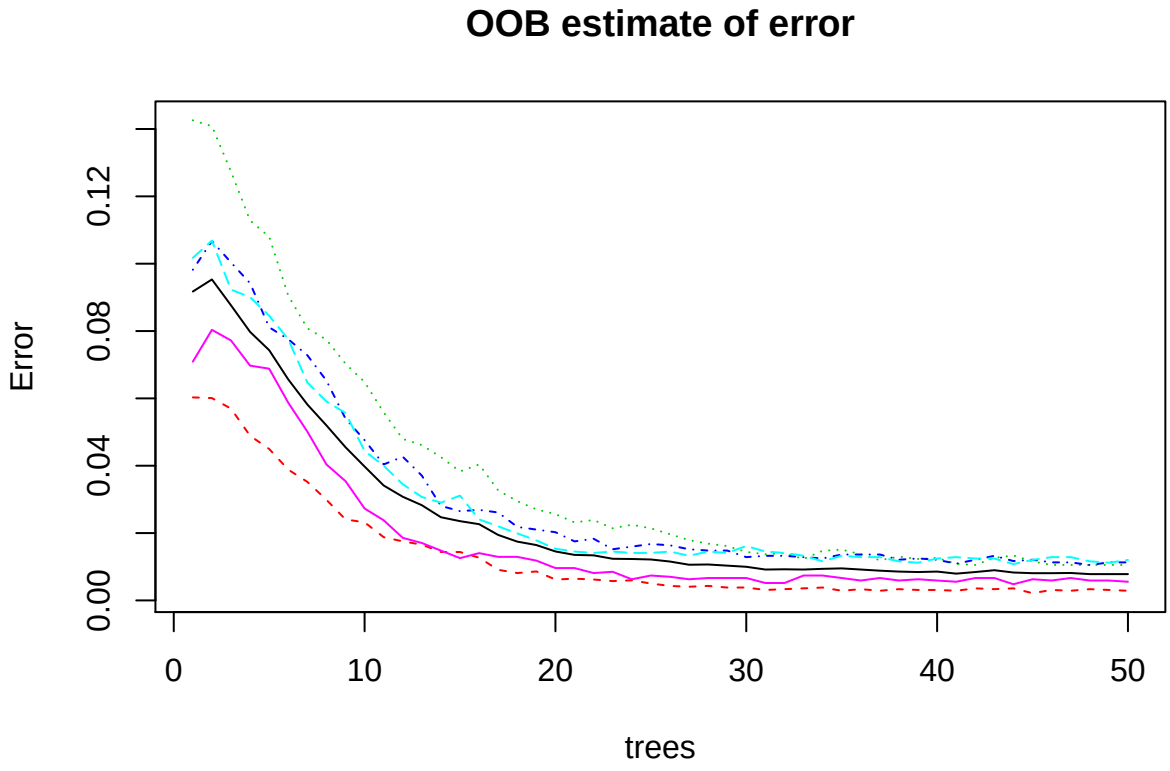
Fitting a Random Forest

To increase the accuracy we instead try a fitting a random forest. In the caret package this is usually done with the method="rf" parameter, however we found the running time of the algorithm to be a bit lackluster. Changing to the randomForest package greatly increases running time.

```
library(randomForest)
rfFit <- randomForest(y=training$classe,x=training[,-53],ntree=50) ## 53 is the classe variable
```

A plot of the out-of-bag error rate

```
plot(rfFit,main="OOB estimate of error")
```



The black line is the OOB error and the coloured lines are the error rates for each of the five classes. The Random Forest model performs well judging from the OOB samples. A relative small number of trees is needed to obtain a high accuracy rating. At 20 trees we have an estimated accuracy of around 98%. How well does the model perform on the testing set?

```
rfPred <- predict(rfFit,newdata=testing)
confMatrix <- confusionMatrix(rfPred,testing$classe)$table
rfAcc <- confusionMatrix(rfPred,testing$classe)$overall
kable(confMatrix)
```

	A	B	C	D	E
A	1395	4	0	0	0
B	0	943	0	0	0
C	0	2	852	6	1
D	0	0	3	797	3
E	0	0	0	1	897

```
rfAcc
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9959217      0.9948411      0.9937084      0.9975071      0.2844617
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

As expected, the random forest performs exceptionally well. We conclude that we are able to classify how well the weight lifting was be performed.