Practical Machine Learning Write-Up: Weight Lifting

Zefanja Kleuters

25 februari 2016

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. 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. In this project, the goal will be 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. Correct and incorrect exercises are classified into A, B, C, D, and E categories.These categories are stored in outcome variable **classe**.

This report describes:

```
- how the model is built,
- how cross validation is performed
- calculation of expected Out of Sample Error
- describe the choices made
- use predition model to predict 20 different test cases
```

Data

The training data for this project are available here: link

The test data are available here: link

The data for this project come from this source: link.

Read and pre processing data

The data training and test data set are read and remove non relevant variables/columns, columns containing mostly NA/empty data and zero value covariates.

```
    # Clear environment, set working directory and read data sets
    rm(list=ls())
    library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

    setwd("C:/Users/Gebruiker/Documents/@Courses/RCode/08
PracticalMachineLearning/PA")
    set.seed(83453)
    trainingData <- read.csv("pml-training.csv", header=TRUE,sep=",",
```

```
na.strings=c("NA","#DIV/0!",""))
    testingData <- read.csv("pml-testing.csv", header=TRUE,sep=",",
na.strings=c("NA","#DIV/0!",""))

    # Preprocessing the data and data clean-up
    dim(trainingData)
```

```
## [1] 19622   160
```

```
    # Line below marked as comment to shorten report
    # str(trainingData)

    # Select only data from the sensors on the belt, forearm, arm and dumbell
and
    # classification on execises execution (classe).

    # Remove the first 6 columns (are informative)
    trainingData <- trainingData[,7:ncol(trainingData)]
    testingData <- testingData[,7:ncol(testingData)]

    # Check columns containing mostly NA
    dblPart <- 0.95
    dim(trainingData[, colSums(is.na(trainingData)) == 0] ) # No NA's
```

```
## [1] 19622    54
```

```
    dim(trainingData[, colSums(is.na(trainingData)) < nrow(trainingData) *
dblPart] ) # 95% NA
```

```
## [1] 19622    54
```

```
    # Line below marked as comment to shorten report
    # colnames(trainingData[, colSums(is.na(trainingData)) <
nrow(trainingData) * dblPart] )

    # Remove columns containing mostly NA
    trainingData <- trainingData[, colSums(is.na(trainingData)) == 0]
    testingData <- testingData[, colSums(is.na(testingData)) == 0]
    dim(trainingData)
```

```
## [1] 19622    54
```

```
    dim(testingData)
```

```
## [1] 20 54
```

```
    # Removing zero value covariates.
    nzvColumns <- nearZeroVar(trainingData) # nzv = TRUE, zeroVar = TRUE or
FALSE

    # nzvColumns is empty so preprocessing is done.
    if(length(nzvColumns) > 0) {
```

```
        trainingData <- trainingData[, -nzvColumns]
        testingData <- testingData[, -nzvColumns]
    }
    dim(trainingData)
```

```
## [1] 19622    54
```

```
    dim(testingData)
```

```
## [1] 20 54
```

Building the model

For cross validation purposes the training data set is randomly split in a training subset (80%) and a test subset (20%). A model is built from the training subset, the test subset will only be used for testing, evaluation and accuracy measurement.

```
    # Breakup the training set in a training and validation data set (80/20).
    # Just to eliminate fitting the model to the testing data.
    inTrain <- createDataPartition(trainingData$classe, p = 0.8, list=FALSE)
    trainSet <- trainingData[ inTrain,]
    testSet <- trainingData[-inTrain,]
```

Random Forest and model simplification by reduction of elements

```
    # Training a model
    library(randomForest)
```

```
## randomForest 4.6-12
```

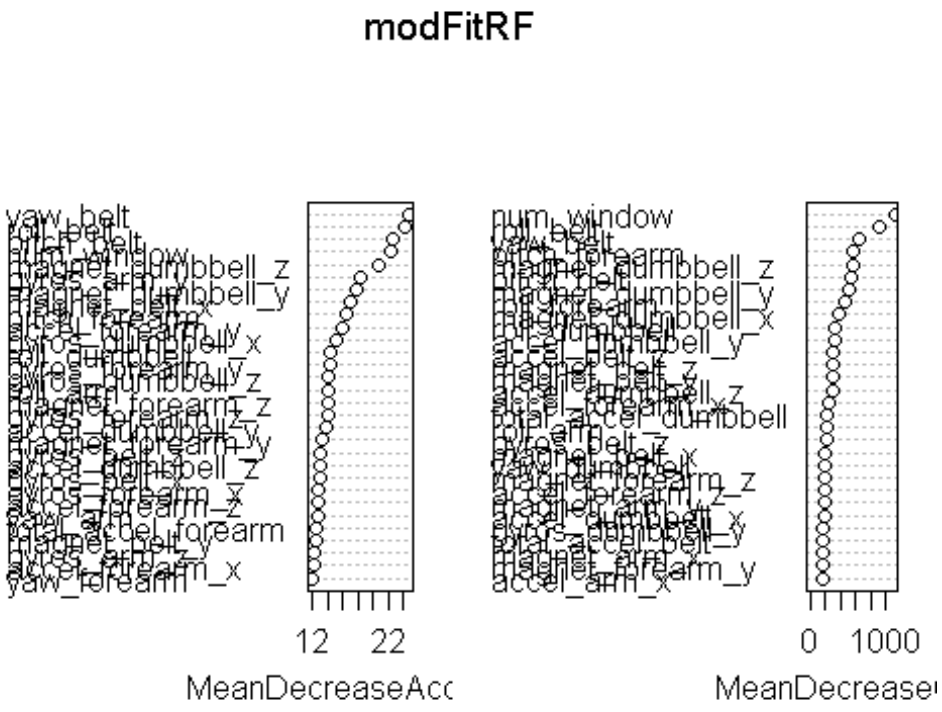
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

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

```
    modFitRF <- randomForest(classe~., data=trainSet, importance=TRUE,
ntree=100)
```

```
    # Variable Importance
    varImpPlot(modFitRF)
```



To reduce the variables in the model, I select the top 10 variables from the Accuracy and Gini graphs above. The reduction of the variables is a good idea when the accuracy of the resulting model is acceptable. The 10 covariates are: "yaw_belt","roll_belt","pitch_belt", "num_window", "magnet_dumbbell_z","pitch_forearm", "magnet_dumbbell_y", "accel_dumbbell_y","roll_arm"and "roll_dumbbell".

```
    # Reduction of elements
    # Correlated predictors.
    predCorrel <- abs(cor(trainSet[,c("yaw_belt","roll_belt","pitch_belt",
"num_window", "magnet_dumbbell_z",
                                      "pitch_forearm", "magnet_dumbbell_y",
"accel_dumbbell_y","roll_arm",
                                      "roll_dumbbell")]))
    diag(predCorrel) <- 0
    which(abs(predCorrel)>0.8, arr.ind=TRUE)
```

```
##           row col
## roll_belt   2   1
## yaw_belt    1   2
```

```
    max(abs(predCorrel))
```

```
## [1] 0.815726
```

Analyzing the correlation between the 10 variables model, there is a high correlation between **roll_belt** and **yaw_belt**. Recalculating the correlation respectively **roll_belt** and **yaw_belt**, the maximum correlation was 0.72 on either. Removing roll_belt left a correlation

of 0.72 between two variables (**roll_dumbbell** and **accel_dumbbell_y**) instead of 4 variables when removing **yaw_belt**.

Building the model

```
    # Building a reduced model with a 2 fold cross validation.
    modFitRF <- train(classe ~ roll_belt + pitch_belt + num_window +
magnet_dumbbell_z + pitch_forearm +
                      magnet_dumbbell_y + accel_dumbbell_y + roll_arm +
roll_dumbbell,
                      data=trainSet, method="rf",
prox=TRUE,allowParallel=TRUE,
                      trControl=trainControl(method="cv",number=2))

    # Complete model below is marked as comment
    # modFitRF <- train(classe~., data=trainSet, method="rf",
prox=TRUE,allowParallel=TRUE,
    #                   trControl=trainControl(method="cv",number=2))
```

Model accuracy and estimation of the out-of-sample error rate

```
    execPredictions <- predict(modFitRF, newdata=testSet)
    confMatrix <- confusionMatrix(execPredictions, testSet$classe)
    confMatrix

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    4    0    0    0
##          B    0  754    2    0    3
##          C    0    0  682    1    0
##          D    0    0    0  641    0
##          E    0    1    0    1  718
##
## Overall Statistics
##
##                Accuracy : 0.9969
##                  95% CI : (0.9947, 0.9984)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9961
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9934   0.9971   0.9969   0.9958
## Specificity            0.9986   0.9984   0.9997   1.0000   0.9994
## Pos Pred Value         0.9964   0.9934   0.9985   1.0000   0.9972
```

```
## Neg Pred Value         1.0000   0.9984   0.9994   0.9994   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1922   0.1738   0.1634   0.1830
## Detection Prevalence   0.2855   0.1935   0.1741   0.1634   0.1835
## Balanced Accuracy      0.9993   0.9959   0.9984   0.9984   0.9976
```

The accuracy of the model with reduces variables is **99,77%**. This accuracy justify using a model with reduced variables.

```
    # Estimation of the out-of-sample error rate
    errRate <- sum(testSet$classe != execPredictions) /
length(testSet$classe)
    errRate

## [1] 0.003058884
```

The out-of-sample error rate is 0.0030589.

Prediction of the testing data

```
    testPredictions <- predict(modFitRF, newdata=testingData)
    testingData$classe <- testPredictions
    table(testingData$problem_id,testingData$classe)

##
##      A B C D E
##   1  0 1 0 0 0
##   2  1 0 0 0 0
##   3  0 1 0 0 0
##   4  1 0 0 0 0
##   5  1 0 0 0 0
##   6  0 0 0 0 1
##   7  0 0 0 1 0
##   8  0 1 0 0 0
##   9  1 0 0 0 0
##   10 1 0 0 0 0
##   11 0 1 0 0 0
##   12 0 0 1 0 0
##   13 0 1 0 0 0
##   14 1 0 0 0 0
##   15 0 0 0 0 1
##   16 0 0 0 0 1
##   17 1 0 0 0 0
##   18 0 1 0 0 0
##   19 0 1 0 0 0
##   20 0 1 0 0 0
```