Practical Machine Learning: Course Project

Practical Machine Learning: Course Project

Background

Six young healthy male participants were asked to perform one set of 10 repitions of the Unilateral Dumpbell Biceps Curl in five different fashions: exactly according to the specification (Class A) , throwing the elbows to the front (Class B), lifting the dumbbell only halfway (class C), lowering the dumbell only halfway (Class D) and throwing the hips to the front (class E). The goal of this project is to identify the execution type of the exercise by the information given from motion sensors on participants' bodies.

Getting data and preprocessing

First I have to read in my data saved in my working directory, unifying the appearance of the "NA"-variables by converting empty strings as well as excel division error strings (#DIV/0!) in "NA"-variables:

```
training<-read.csv("pml-training.csv", na.strings=c("","#DIV/0!","NA"))
testing<-read.csv("pml-testing.csv", na.strings=c("","#DIV/0!","NA"))
dim(training)
```

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

```
dim(testing)
```

```
## [1]  20 160
```

Both datasets contain the same variables (with exception from the last, which is "classe" in "training" and "problem_id" in "testing"):

Typing str(training) reveals that the first seven columns are irrelevant as predictors, so we set:

```
training<-training[,-c(1:7)]
```

Also there seem to be variables with lots of “NA”. I decide to not include those variables with more than 80% “NA”-entries as predictors:

```
c<-matrix(153,1)
for(i in 1:153)
{c[i]=ifelse( sum(is.na(training[,c(i)]))>0.8*19622,i,0)}
print(c)
```

```
##   [1]   0   0   0   0   5   6   7   8   9  10  11  12  13  14  15  16  17
##  [18]  18  19  20  21  22  23  24  25  26  27  28  29   0   0   0   0   0
##  [35]   0   0   0   0   0   0   0   0  43  44  45  46  47  48  49  50  51
##  [52]  52   0   0   0   0   0   0   0   0   0  62  63  64  65  66  67  68
##  [69]  69  70  71  72  73  74  75  76   0   0   0  80  81  82  83  84  85
##  [86]  86  87  88  89  90  91  92  93  94   0  96  97  98  99 100 101 102
## [103] 103 104 105   0   0   0   0   0   0   0   0   0   0   0   0 118 119
## [120] 120 121 122 123 124 125 126 127 128 129 130 131 132   0 134 135 136
## [137] 137 138 139 140 141 142 143   0   0   0   0   0   0   0   0   0   0
```

This leads to

```
training<-training[,-c(5:29,43:52,62:76,80:94,96:105,118:132,134:143)]
```

Maybe we could reduce further prediction variables:

```
library(caret)
nzv<-subset(nearZeroVar(training, saveMetrics=TRUE),zeroVar==TRUE |nzv==TRUE)
nzv
```

```
## [1] freqRatio     percentUnique zeroVar       nzv
## <0 rows> (or 0-length row.names)
```

This is obviously not the case.So we get the following prediction variables:

```
names(training[,-c(53)])
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"
```

We subdivide the training set into a train section and a test section:

```
library(caret)
set.seed(36546)
trainingIndex  <- createDataPartition(training$classe, p=.60, list=FALSE)
training.train <- training[ trainingIndex,]
training.test  <- training[-trainingIndex,]
```

As I now have divided my already classified data in a train and a test part, I would like to apply some prediction models and compare them to each other.

Prediction with k-Nearest Neighbour Classification

```
library(kknn)
set.seed(36546)
ModelKKNN<-kknn(classe~.,training.train,training.test)
```

X contains the predictions for the variable classe for training.test by the k-Nearest Neighbour Algorithm:

```
X<-as.factor(ModelKKNN$CL[,1])
```

Y contains the variable "classe" from the training.test dataset:

```
Y<-training.test$classe
```

This leads to the following Confusion Matrix:

```
cm.KKNN<-confusionMatrix(X,Y)
cm.KKNN
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2224   15    4    3    2
##          B    5 1475   14    0    5
##          C    2   25 1332   15    1
##          D    1    2   18 1267    6
##          E    0    1    0    1 1428
##
## Overall Statistics
##
##                Accuracy : 0.9847
##                  95% CI : (0.9817, 0.9873)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9807
##  Mcnemar's Test P-Value : 0.01922
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9717   0.9737   0.9852   0.9903
## Specificity            0.9957   0.9962   0.9934   0.9959   0.9997
## Pos Pred Value         0.9893   0.9840   0.9687   0.9791   0.9986
## Neg Pred Value         0.9986   0.9932   0.9944   0.9971   0.9978
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2835   0.1880   0.1698   0.1615   0.1820
## Detection Prevalence   0.2865   0.1911   0.1752   0.1649   0.1823
## Balanced Accuracy      0.9961   0.9839   0.9835   0.9906   0.9950
```

Prediction with Decision Trees

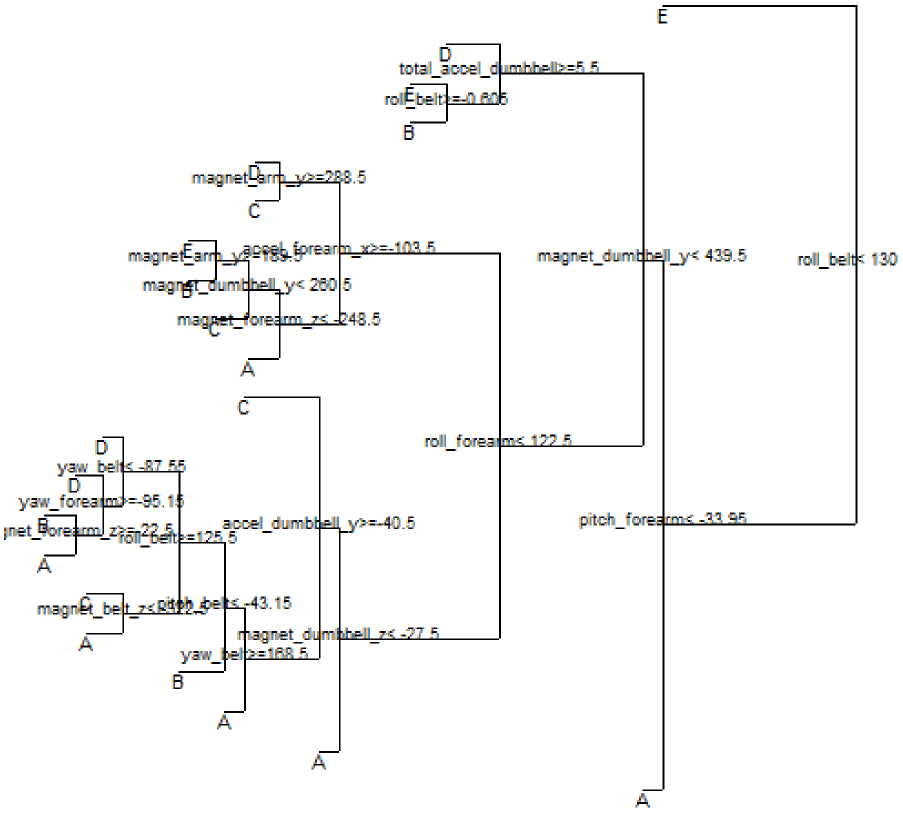
```
library(rpart)
library(rpart.plot)
set.seed(36546)
modelRPART<-rpart(classe ~ ., data=training.train, method="class")
```

Applying our model to the test portion of our training set:

```
predRPART <- predict(modelRPART, training.test, type = "class")
```

Plotting the Decision Tree as a (flipped) dendrogram:

```
library(ggplot2)
library(ggdendro)
ddata <- dendro_data(modelRPART)
g<-ggplot() +
    geom_segment(data = ddata$segments,
                 aes(x = x, y = y, xend = xend, yend = yend)) +
    geom_text(data = ddata$labels,
              aes(x = x, y = y, label = label), size = 3, vjust = 0) +
    geom_text(data = ddata$leaf_labels,
              aes(x = x, y = y, label = label), size = 4, vjust = 1) +
    theme_dendro() + coord_flip()
print(g)
```



This leads to the following Confusion Matrix:

```
cm.RPART <- confusionMatrix(predRPART, training.test$classe)
cm.RPART
```

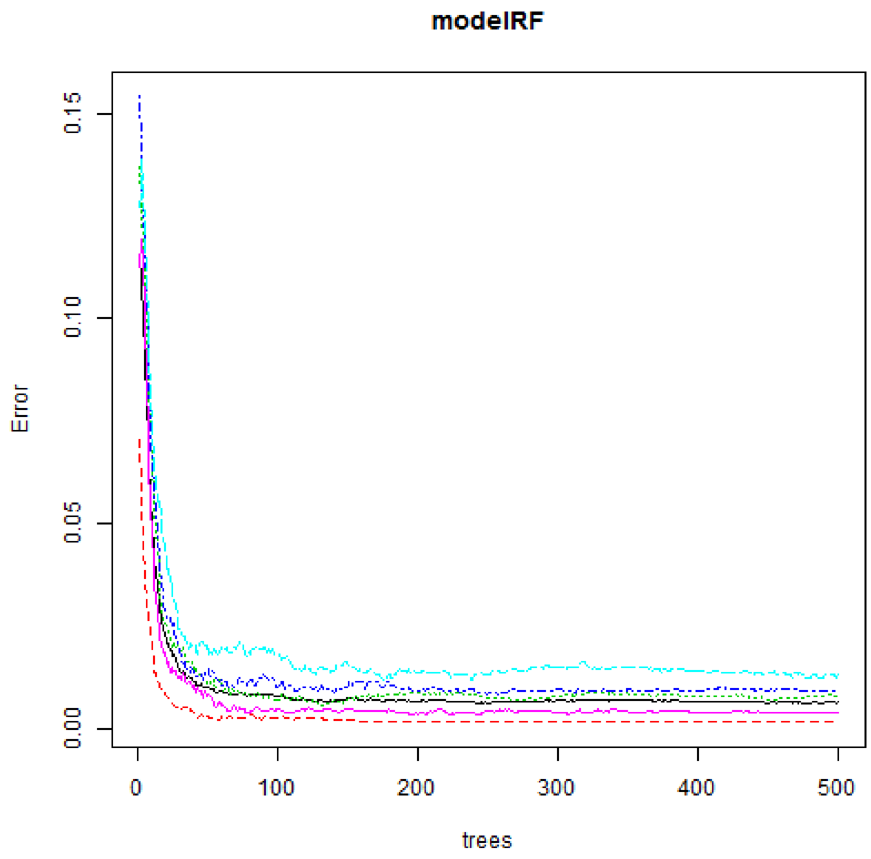
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2003  331   23  116   47
##          B   64  850  215   70  167
##          C   61  178  953  122  100
##          D   93  119  125  907  199
##          E   11   40   52   71  929
##
## Overall Statistics
##
##                Accuracy : 0.7191
##                  95% CI : (0.709, 0.729)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6433
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8974   0.5599   0.6966   0.7053   0.6442
## Specificity            0.9079   0.9185   0.9288   0.9183   0.9728
## Pos Pred Value         0.7948   0.6223   0.6740   0.6286   0.8422
## Neg Pred Value         0.9570   0.8969   0.9355   0.9408   0.9239
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2553   0.1083   0.1215   0.1156   0.1184
## Detection Prevalence   0.3212   0.1741   0.1802   0.1839   0.1406
## Balanced Accuracy      0.9027   0.7392   0.8127   0.8118   0.8085
```

Prediction with Random Forest

```
library(randomForest)
set.seed(36546)
modelRF<-randomForest(classe ~ ., data=training.train)
```

```
## Error: cannot allocate vector of size 89.8 Mb
```

```
predRF <- predict(modelRF, training.test, type = "class")
plot(modelRF)
```



This leads to the following Confusion Matrix:

```
cm.RF <- confusionMatrix(predRF, training.test$classe)
cm.RF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   10    0    0    0
##          B    1 1501   10    0    0
##          C    0    7 1357   14    0
##          D    0    0    1 1272    5
##          E    0    0    0    0 1437
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9939
##                  95% CI : (0.9919, 0.9955)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9888   0.9920   0.9891   0.9965
## Specificity            0.9982   0.9983   0.9968   0.9991   1.0000
## Pos Pred Value         0.9955   0.9927   0.9848   0.9953   1.0000
## Neg Pred Value         0.9998   0.9973   0.9983   0.9979   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1913   0.1730   0.1621   0.1832
## Detection Prevalence   0.2856   0.1927   0.1756   0.1629   0.1832
## Balanced Accuracy      0.9989   0.9935   0.9944   0.9941   0.9983
```

Comparing of the Different Methods

Compared with the other methods, the prediction with Decision trees results in a big out-of-sample-error (1-accuracy). The accuracy for Random Forest Prediction is very high (0.9939), even higher than the one for k-Nearest Neighbour Classification (0.9847), so we use it for our testing set:

```
pred<- predict(modelRF,testing,type="class")
pred
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```