Categorizing weightlifting exercises

Michel MariÃ«n

18 september 2018

Introduction

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell 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 dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience.

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Setup

```
library(ggplot2)
library(GGally)
library(reshape2)
library(AppliedPredictiveModeling)
library(dplyr)
library(MASS)
library(caret)
library(Hmisc)
library(ElemStatLearn)
library(pgmm)
library(rpart)
library(rattle)
library(randomForest)
library(gridExtra)
library(devtools)
library(janitor)
library(tidyr)

setwd("C:\\_Data\\Mijn Documenten\\R\\Data Science specialization\\Scripts Courses\\Course 8 - Week 4")

dataset_train<-read.csv("pml-training.csv",dec=".",fill=TRUE,stringsAsFactors = FALSE)
dataset_test<-read.csv("pml-testing.csv",dec=".",fill=TRUE,stringsAsFactors = FALSE)

data_train<-dataset_train
data_test<-dataset_test
```

Tidying dataset

Before any attempt is made to create a machine learning-model, the data is first cleaned. To make sure all data is numeric after import into R, all columns are converted to numeric, the categories and names are converted to factor and the cvtd-columns is converted to POSIXct.

After that, all columns containing "NA" or are empty are removed from the test and training set.

```
  ## Kolommen naar juiste klasses zetten
    data_train[,-c(2:6,160)] <- sapply( data_train[,-c(2:6,160)], as.numeric )
    data_train[,2] <- as.factor(data_train[,2])
    data_train[,6]<- as.factor(data_train[,6])
    data_train[,160]<- as.factor(data_train[,160])
    data_train[,5]<- as.POSIXct(data_train[,5],format="%d/%m/%Y %H:%M",tz="GMT")

    data_test[,-c(2:6,160)] <- sapply( data_test[,-c(2:6,160)], as.numeric )
    data_test[,2] <- as.factor(data_test[,2])
    data_test[,6]<- as.factor(data_test[,6])
    data_test[,160]<- as.factor(data_test[,160])
    data_test[,5]<- as.POSIXct(data_test[,5],format="%d/%m/%Y %H:%M",tz="GMT")

    hh<-lapply(data_train,class)
    table(unlist(hh))
```

```
##
##  factor integer numeric POSIXct  POSIXt
##       3       2     154       1       1
```

```
  ## Verwijderen kolommen met NA-waardes
    Na_rows<-as.data.frame(sapply(data_train, function(x) sum(is.na(x))))
    Na_rows$Namen<-rownames(Na_rows)
    Na_rows<-Na_rows[Na_rows[,1]==0,]

    data_tr<-subset(data_train,select=Na_rows[,2])
    data_te<-subset(data_test,select=Na_rows[c(1:59),2])
    data_te<-data_te%>%mutate(ToPredict=data_test[,160])

    data_tr<-data_tr[,-c(1:5)]
    data_te<-data_te[,-c(1:5)]
```

Exploratory analysis

The dataset with 54 variables is to big to visualize. To get a sense of the kind of data we are working with, the describe function is applied to the data. The output is explicitly excluded from this document since it contains a pretty big table.

```
   describe(data_tr)
```

Training

For selecting the best possible model to classify the excersises, 4 different models will be trained and, depending on their accuracy and Kappa-values, the best will be chosen.

```
  ## Create training dataset
    set.seed(555)
    inTrain = createDataPartition(data_tr$classe, p = 0.7,list=FALSE)
    training = data_tr[inTrain,]
    testing = data_tr[-inTrain,]

    fitControl <- trainControl(method = "cv", number=3)

  ## Decision tree
    Model_dtr<-train(classe~.,method="rpart",data=training,trControl=fitControl)
    Model_dtr$finalModel
```

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##   2) roll_belt< 129.5 12496 8640 A (0.31 0.21 0.19 0.18 0.11) *
##   3) roll_belt>=129.5 1241   50 E (0.04 0 0 0 0.96) *
```

```
  ## Random forest
    Model_rfo<-train(classe~.,method="rf",data=training,trControl=fitControl)
    Model_rfo$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 28
##
##         OOB estimate of  error rate: 0.19%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3906    0    0    0    0 0.000000000
## B    6 2650    2    0    0 0.003009782
## C    0    4 2391    1    0 0.002086811
## D    0    0   10 2242    0 0.004440497
## E    0    0    0    3 2522 0.001188119
```

```
  ## Linear discriminat analysis
    Model_lda<-train(classe~.,method="lda",data=training,trControl=fitControl)
    Model_lda$finalModel
```

```
## Call:
## lda(x, grouping = y)
##
## Prior probabilities of groups:
##         A         B         C         D         E
## 0.2843416 0.1934920 0.1744195 0.1639368 0.1838101
##
```

```
## Group means:
##   new_windowyes num_window roll_belt  pitch_belt   yaw_belt
## A    0.02022529   382.9834  59.11012  0.25642857 -12.064887
## B    0.02144470   502.4759  65.14494 -0.04406321 -13.884718
## C    0.02086811   487.2538  64.30970 -1.04746661  -7.889983
## D    0.02220249   431.6781  60.94462  1.72952487 -17.553131
## E    0.02376238   373.4455  74.26392  0.55533465  -5.532166
##   total_accel_belt gyros_belt_x gyros_belt_y gyros_belt_z accel_belt_x
## A         10.65463 -0.004009217   0.04053251   -0.1208628    -6.108295
## B         11.12829 -0.006136193   0.04256584   -0.1349624    -4.899925
## C         11.11895 -0.013626878   0.03969533   -0.1332220    -4.034641
## D         11.22069 -0.014888988   0.03682504   -0.1348268    -8.086146
## E         12.65109  0.010499010   0.03818614   -0.1295683    -4.456634
##   accel_belt_y accel_belt_z magnet_belt_x magnet_belt_y magnet_belt_z
## A     28.72888    -62.17256      57.87634      602.4409     -337.5635
## B     31.96313    -73.68284      49.30662      598.9176     -337.6027
## C     30.85392    -70.26669      57.03798      600.2237     -336.4754
## D     30.46803    -69.53863      49.01865      593.8601     -341.4516
## E     28.52000    -91.32158      62.96158      568.8642     -377.6329
##     roll_arm  pitch_arm    yaw_arm total_accel_arm gyros_arm_x gyros_arm_y
## A -0.1142473   3.156436 -11.974736        27.48618 0.007918587  -0.2129544
## B 32.0520166  -6.113040   7.844105        26.58766 0.006471031  -0.2774266
## C 25.2133431  -1.566732   5.425472        24.06678 0.132257930  -0.2762896
## D 22.8854130 -10.250306   4.944987        23.38144 0.030510657  -0.2535879
## E 19.7320911 -12.218982  -2.500139        24.50931 0.022724752  -0.2642139
##   gyros_arm_z accel_arm_x accel_arm_y accel_arm_z magnet_arm_x
## A   0.2606528  -131.62289    47.40988   -75.97773    -15.50691
## B   0.2644582   -44.15124    26.09744   -94.08804    228.68886
## C   0.2827129   -78.40943    41.80342   -52.57554    159.52504
## D   0.2713055    14.89210    25.48179   -46.35924    396.48934
## E   0.2760911   -20.55366    16.59564   -75.87525    320.74139
##   magnet_arm_y magnet_arm_z roll_dumbbell pitch_dumbbell yaw_dumbbell
## A    234.86329     408.2391      21.03830     -19.422425    0.4168088
## B    132.46764     199.1418      34.85467       3.035460   15.1755999
## C    191.86561     366.2166     -13.57914     -25.007881  -15.8625938
## D     97.75799     300.7269      50.77625      -1.981835    1.2407902
## E     85.26574     220.5945      26.60479      -7.034579    5.5159257
##   total_accel_dumbbell gyros_dumbbell_x gyros_dumbbell_y gyros_dumbbell_z
## A             14.77675        0.1689068      0.021413210       -0.1553943
## B             14.18924        0.1715388      0.006647856       -0.1434274
## C             12.90109        0.1932721      0.052921536       -0.1521035
## D             11.28996        0.2042895      0.013379218       -0.1308082
## E             14.48158        0.1326099      0.115805941       -0.1435485
##   accel_dumbbell_x accel_dumbbell_y accel_dumbbell_z magnet_dumbbell_x
## A      -51.1886841         53.11905        -58.45929         -389.7752
## B       -0.3871332         67.50677        -14.88826         -249.4816
## C      -40.4177796         30.73205        -52.67571         -374.1903
## D      -21.9094139         53.06972        -32.93295         -315.8641
## E      -18.0550495         56.47089        -24.67842         -291.3648
##   magnet_dumbbell_y magnet_dumbbell_z roll_forearm pitch_forearm
## A          219.4160          10.83359     26.18179      -6.89905
## B          265.0764          50.57562     31.04043      14.58393
## C          161.5346          61.86394     59.28809      12.34838
## D          218.0844          58.12567     14.86397      27.97656
```

```
## E          242.6004          72.74970     39.97962      16.78754
##   yaw_forearm total_accel_forearm gyros_forearm_x gyros_forearm_y
## A   24.847343            32.19892       0.1803328      0.07987711
## B   12.473593            35.41798       0.1452596      0.06277652
## C   39.216244            35.00292       0.2108431      0.04193239
## D    3.835195            36.09236       0.1215275     -0.02034192
## E   12.261513            36.79881       0.1252673      0.10796436
##   gyros_forearm_z accel_forearm_x accel_forearm_y accel_forearm_z
## A       0.1168228      -0.8832565        169.1582       -59.95366
## B       0.1740858     -76.6422122        137.9251       -44.56697
## C       0.1407846     -49.9023372        211.2617       -63.47078
## D       0.1152664    -153.1936057        153.8890       -46.41607
## E       0.1578178     -72.4170297        146.2701       -57.88515
##   magnet_forearm_x magnet_forearm_y magnet_forearm_z
## A        -195.3989         469.5064         409.2148
## B        -325.9620         280.6629         372.5256
## C        -338.8088         501.1540         463.3080
## D        -452.6190         317.9050         359.8908
## E        -335.4277         285.2028         352.2729
##
## Coefficients of linear discriminants:
##                               LD1           LD2           LD3
## new_windowyes         5.600814e-02  0.1068907068  0.0149664730
## num_window            4.682913e-04 -0.0008133960  0.0016150411
## roll_belt             5.668576e-02  0.0942601405  0.0136081736
## pitch_belt            3.315691e-02  0.0146043866 -0.0707167615
## yaw_belt             -8.717945e-03  0.0009254081 -0.0100555794
## total_accel_belt     -4.462441e-02  0.0006375498 -0.2946817841
## gyros_belt_x          7.057177e-01 -0.0277058230  0.9920597093
## gyros_belt_y         -1.744124e+00 -2.1638341564 -1.1519697215
## gyros_belt_z          6.478435e-01  0.4343620021  0.5247307731
## accel_belt_x         -1.492112e-03 -0.0008116721  0.0203003890
## accel_belt_y         -2.770221e-02 -0.0409552147  0.0531220135
## accel_belt_z          2.947556e-03  0.0262614182 -0.0059616763
## magnet_belt_x        -1.190060e-02  0.0038318177 -0.0213051237
## magnet_belt_y        -2.116994e-02 -0.0065029601 -0.0015797767
## magnet_belt_z         7.594111e-03 -0.0013569960  0.0113130857
## roll_arm              6.833165e-04  0.0001941309  0.0021953162
## pitch_arm            -3.030386e-03  0.0058660705  0.0057989684
## yaw_arm               1.168253e-03 -0.0009505459  0.0015729853
## total_accel_arm       3.469824e-03 -0.0209156060 -0.0232186351
## gyros_arm_x           1.243555e-01  0.0381250380 -0.0778783478
## gyros_arm_y           8.835566e-02 -0.0257321398 -0.1726302054
## gyros_arm_z          -1.164198e-01 -0.1855049761 -0.0357956755
## accel_arm_x          -3.336514e-03 -0.0038706472 -0.0085335171
## accel_arm_y          -3.428814e-03  0.0147194910  0.0006329019
## accel_arm_z           9.881619e-03 -0.0019641372  0.0017550750
## magnet_arm_x          1.515097e-04 -0.0006097083  0.0019929055
## magnet_arm_y         -9.498633e-04 -0.0055777618  0.0046707724
## magnet_arm_z         -3.923660e-03 -0.0017895596 -0.0057559904
## roll_dumbbell         2.444041e-03 -0.0038951349 -0.0028697401
## pitch_dumbbell       -5.736338e-03 -0.0032133171 -0.0042536551
## yaw_dumbbell         -7.860023e-03  0.0073569263 -0.0028826148
## total_accel_dumbbell  7.059123e-02  0.0637671088  0.0059682772
```

```
## gyros_dumbbell_x      2.990193e-01 -0.4974424627  0.1943731392
## gyros_dumbbell_y      1.992857e-01 -0.2759412578 -0.0101327546
## gyros_dumbbell_z      2.430631e-01 -0.3426717472 -0.0471057181
## accel_dumbbell_x      1.280482e-02  0.0083767216  0.0015985002
## accel_dumbbell_y      2.030489e-03  0.0028573906  0.0016037522
## accel_dumbbell_z      2.612248e-03  0.0018305476  0.0022538971
## magnet_dumbbell_x    -4.006293e-03 -0.0004903687  0.0033762134
## magnet_dumbbell_y    -1.158856e-03  0.0025262623 -0.0004872572
## magnet_dumbbell_z     1.329573e-02 -0.0097344850 -0.0019944205
## roll_forearm          1.500661e-03  0.0012797531  0.0001630616
## pitch_forearm         1.693030e-02 -0.0135380645  0.0043764687
## yaw_forearm          -3.231909e-05  0.0008050417  0.0007071903
## total_accel_forearm   3.228019e-02  0.0059892633 -0.0057270964
## gyros_forearm_x      -7.325181e-02 -0.0702389205  0.1932342764
## gyros_forearm_y      -2.411620e-02 -0.0249028963  0.0149596114
## gyros_forearm_z       1.029859e-01  0.1060647735 -0.0667830920
## accel_forearm_x       3.437968e-03  0.0104763489  0.0009959472
## accel_forearm_y       6.105396e-04 -0.0008748640 -0.0008885443
## accel_forearm_z      -7.176906e-03  0.0026778164  0.0038177547
## magnet_forearm_x     -1.732973e-03 -0.0034817976 -0.0001982994
## magnet_forearm_y     -8.540197e-04 -0.0014991041  0.0003218647
## magnet_forearm_z     -9.313238e-05 -0.0014521720 -0.0003886225
##                             LD4
## new_windowyes        -6.771551e-02
## num_window            1.139107e-05
## roll_belt             7.240317e-02
## pitch_belt            1.138754e-02
## yaw_belt             -2.917749e-03
## total_accel_belt     -1.636427e-01
## gyros_belt_x          4.333416e-01
## gyros_belt_y          1.101319e+00
## gyros_belt_z         -6.678523e-01
## accel_belt_x          3.687320e-03
## accel_belt_y          4.708020e-03
## accel_belt_z          1.810631e-02
## magnet_belt_x        -3.900713e-03
## magnet_belt_y        -4.603781e-03
## magnet_belt_z         3.024298e-03
## roll_arm              4.122151e-04
## pitch_arm             1.661043e-03
## yaw_arm              -1.225131e-03
## total_accel_arm      -2.019805e-02
## gyros_arm_x           5.483058e-02
## gyros_arm_y           2.102936e-01
## gyros_arm_z           1.348140e-01
## accel_arm_x          -2.203611e-03
## accel_arm_y           3.214921e-03
## accel_arm_z          -7.612257e-03
## magnet_arm_x          1.362055e-03
## magnet_arm_y          7.557131e-04
## magnet_arm_z          2.162059e-03
## roll_dumbbell        -8.003495e-03
## pitch_dumbbell       -4.295022e-03
## yaw_dumbbell         -3.151367e-03
```

```
## total_accel_dumbbell  5.652590e-03
## gyros_dumbbell_x      5.191695e-02
## gyros_dumbbell_y      2.081384e-01
## gyros_dumbbell_z      3.162848e-02
## accel_dumbbell_x      6.697550e-03
## accel_dumbbell_y     -1.071147e-03
## accel_dumbbell_z      1.333693e-03
## magnet_dumbbell_x    -2.695373e-03
## magnet_dumbbell_y    -2.425973e-03
## magnet_dumbbell_z     9.411578e-03
## roll_forearm          1.287373e-03
## pitch_forearm        -6.661253e-04
## yaw_forearm           1.111131e-03
## total_accel_forearm   2.800586e-03
## gyros_forearm_x       1.218344e-01
## gyros_forearm_y      -8.819375e-05
## gyros_forearm_z      -1.468135e-02
## accel_forearm_x       3.633198e-03
## accel_forearm_y      -2.134908e-03
## accel_forearm_z      -4.711051e-03
## magnet_forearm_x     -1.139978e-03
## magnet_forearm_y      3.706136e-04
## magnet_forearm_z      1.136220e-03
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4
## 0.4763 0.2419 0.1701 0.1117
```

```
  ## kNN-model
    Model_knn<-train(classe~.,method="knn",data=training,trControl=fitControl)
    Model_knn$finalModel
```

```
## 5-nearest neighbor model
## Training set outcome distribution:
##
##    A    B    C    D    E
## 3906 2658 2396 2252 2525
```

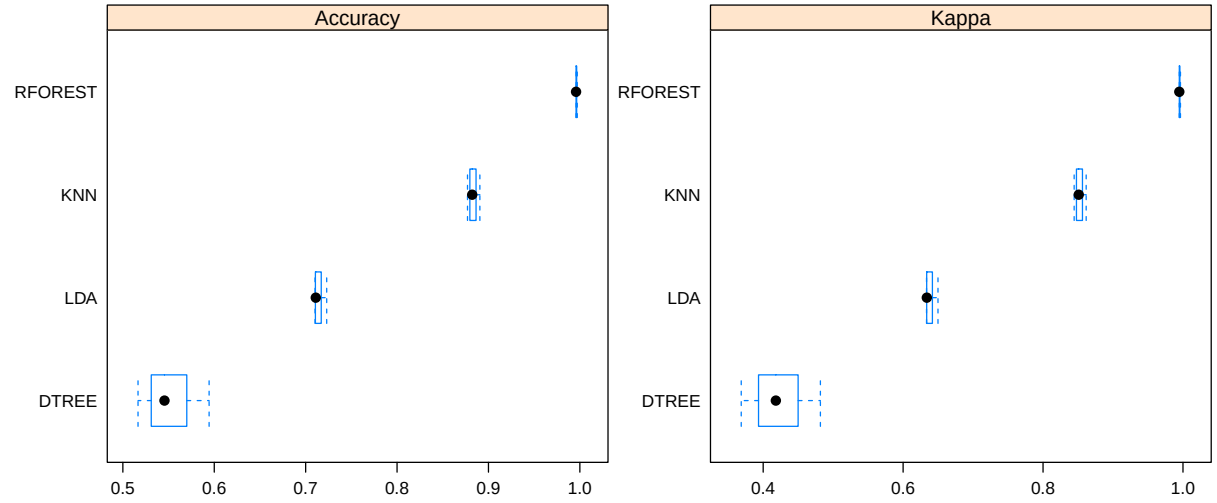
```
  ## Group results
    Model_results <- resamples(list(DTREE=Model_dtr, LDA=Model_lda, KNN=Model_knn, RFOREST=Model_rfo))

    summary(Model_results)
```

```
##
## Call:
## summary.resamples(object = Model_results)
##
## Models: DTREE, LDA, KNN, RFOREST
## Number of resamples: 3
##
## Accuracy
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## DTREE   0.5165939 0.5310639 0.5455340 0.5521641 0.5699492 0.5943644    0
```

```
## LDA      0.7102620 0.7106671 0.7110723 0.7147858 0.7170477 0.7230232    0
## KNN      0.8770742 0.8796558 0.8822373 0.8833076 0.8864243 0.8906114    0
## RFOREST 0.9958497 0.9958502 0.9958506 0.9962873 0.9965061 0.9971616    0
##
## Kappa
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## DTREE    0.3686213 0.3933636 0.4181060 0.4228345 0.4499411 0.4817763    0
## LDA      0.6334626 0.6336764 0.6338903 0.6390604 0.6418593 0.6498283    0
## KNN      0.8444621 0.8477343 0.8510065 0.8523714 0.8563261 0.8616457    0
## RFOREST 0.9947496 0.9947503 0.9947509 0.9953035 0.9955804 0.9964099    0
```

```
    scales<-list(x=list(relation="free"),y=list(relation="free"))
    bwplot(Model_results,scales=scales)
```




```
  ## Predict on test-data
    Model_dtr_predict<-predict(Model_dtr, newdata=testing[,-60])
    Model_lda_predict<-predict(Model_lda, newdata=testing[,-60])
    Model_knn_predict<-predict(Model_knn, newdata=testing[,-60])
    Model_rfo_predict<-predict(Model_rfo, newdata=testing[,-60])

    cfm_dtr<-confusionMatrix(testing$classe, Model_dtr_predict)
    cfm_lda<-confusionMatrix(testing$classe, Model_lda_predict)
    cfm_knn<-confusionMatrix(testing$classe, Model_knn_predict)
    cfm_rfo<-confusionMatrix(testing$classe, Model_rfo_predict)

    matrix_accu<-rbind(cfm_dtr$overall,cfm_lda$overall,cfm_knn$overall,cfm_rfo$overall)
    rownames(matrix_accu)<-c("Tree","LDA","KNN","RFO")
    matrix_accu
```

```
##       Accuracy     Kappa AccuracyLower AccuracyUpper AccuracyNull
## Tree 0.3666950 0.1260439     0.3543665     0.3791558    0.9096007
## LDA  0.7155480 0.6396298     0.7038316     0.7270504    0.3038233
## KNN  0.9221750 0.9015104     0.9150342     0.9288939    0.2912489
## RFO  0.9972812 0.9965609     0.9955886     0.9984452    0.2847918
```

```
##      AccuracyPValue McnemarPValue
## Tree              1           NaN
## LDA               0  8.394111e-52
## KNN               0  2.446864e-13
## RFO               0           NaN
```

```
    OoS_error<-mean(testing$classe != Model_rfo_predict)
    OoS_error
```

```
## [1] 0.002718777
```

From the code above, one can see that the random forest-model has the highest accuracy (0.997) and Kappa-value (0.997). The Out-of-Sample error for the random forest-model (OoS_error) is 0.0027 which is very low and should no wrong classifications on the validation set. This means that for the final prediction, the "Model_rfo_predict" will be used

Prediction on validation-data

Applying our Random Forest-model on the validation-set

```
    Model_rfo_predict_final<-predict(Model_rfo, newdata=data_test)
    Model_rfo_predict_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
```