Practical Machine Learning

Prediction Assignment Write-up

===

Run time: 2021-01-28 17:00:49
R version: R version 4.0.3 (2020-10-10)

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 type 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. In this project, your 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. 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).

Data

The training data for this project are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Prepare the datasets

Transforming the training data into a data table:

```
require(data.table)
```

```
## Loading required package: data.table
```

```
url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
D <- fread(url)
```

Transforming the testing data into a data table:

```
url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
DTest <- fread(url)
```

The variables that do not have any missing values in the test dataset, will be **predictor candidates** and these variables are: belt, arm, dumbbell & forearm.

```
isAnyMissing <- sapply(DTest, function (x) any(is.na(x) | x == ""))
isPredictor <- !isAnyMissing & grepl("belt|[^(fore)]arm|dumbbell|forearm", names(isAnyMissing))
predCandidates <- names(isAnyMissing)[isPredictor]
predCandidates
```

```
##  [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"
```

Sub-setting the primary dataset in such a way, so to include only the **predictor candidates** and the classe outcome variable.

```
varToInclude <- c("classe", predCandidates)
D <- D[, varToInclude, with=FALSE]
dim(D)
```

```
## [1] 19622    53
```

```
names(D)
```

```
##  [1] "classe"               "roll_belt"            "pitch_belt"
##  [4] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
##  [7] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [10] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [13] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [16] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [28] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [31] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [34] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
```

```
## [37] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [40] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [43] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [46] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [49] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [52] "magnet_forearm_y"     "magnet_forearm_z"
```

Making the outcome variable, `classe`, into a factor:

```
D <- D[, classe := factor(D[, classe])]
D[, .N, classe]
```

```
##    classe    N
## 1:      A 5580
## 2:      B 3797
## 3:      C 3422
## 4:      D 3216
## 5:      E 3607
```

Splitting the dataset into: 60% training and 40% probing:

```
require(caret)
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
seed <- as.numeric(as.Date("2021-01-28"))
set.seed(seed)
inTrain <- createDataPartition(D$classe, p=0.6)
DTrain <- D[inTrain[[1]]]
DProbe <- D[-inTrain[[1]]]
```

Reprocessing the prediciton variables, using centering and scaling:

```
X <- DTrain[, predCandidates, with=FALSE]
preProc <- preProcess(X)
preProc
```

```
## Created from 11776 samples and 52 variables
##
## Pre-processing:
##   - centered (52)
##   - ignored (0)
##   - scaled (52)
```

```
XCS <- predict(preProc, X)
DTrainCS <- data.table(data.frame(classe = DTrain[, classe], XCS))
```

Applying centering and scaling to the probing dataset:

```
X <- DProbe[, predCandidates, with=FALSE]
XCS <- predict(preProc, X)
DProbeCS <- data.table(data.frame(classe = DProbe[, classe], XCS))
```

Checking whether there is near zero variance:

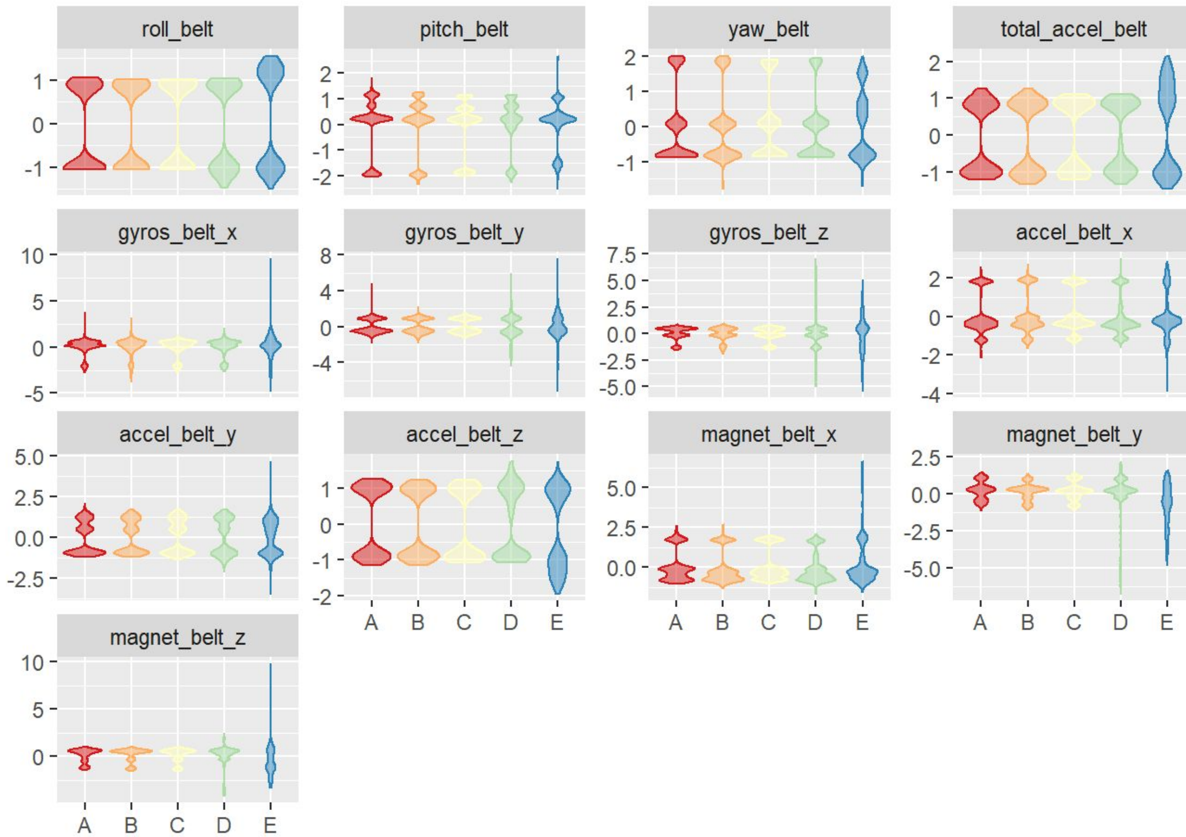
```
nzv <- nearZeroVar(DTrainCS, saveMetrics=TRUE)
if (any(nzv$nzv)) nzv else message("No variables with near zero variance")
```

```
## No variables with near zero variance
```

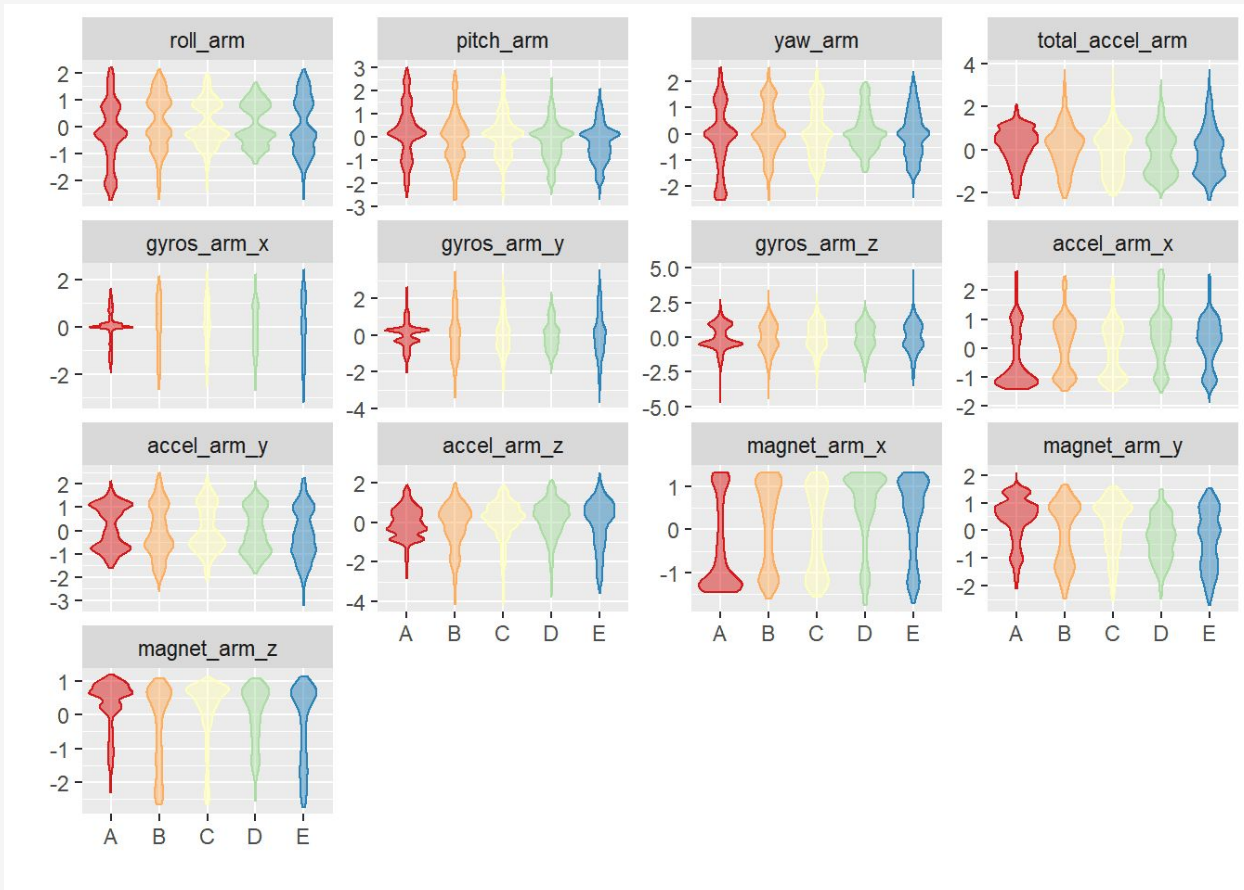
Examining the groups of prediction variables:

```
histGroup <- function (data, regex) {
  col <- grep(regex, names(data))
  col <- c(col, which(names(data) == "classe"))
  require(reshape2)
  n <- nrow(data)
  DMelted <- melt(data[, col, with=FALSE][, rownum := seq(1, n)], id.vars=c("rownum", "classe"))
  require(ggplot2)
  ggplot(DMelted, aes(x=classe, y=value)) +
    geom_violin(aes(color=classe, fill=classe), alpha=1/2) +
#   geom_jitter(aes(color=classe, fill=classe), alpha=1/10) +
#   geom_smooth(aes(group=1), method="gam", color="black", alpha=1/2, size=2) +
    facet_wrap(~ variable, scale="free_y") +
    scale_color_brewer(palette="Spectral") +
    scale_fill_brewer(palette="Spectral") +
    labs(x="", y="") +
    theme(legend.position="none")
}
histGroup(DTrainCS, "belt")
```

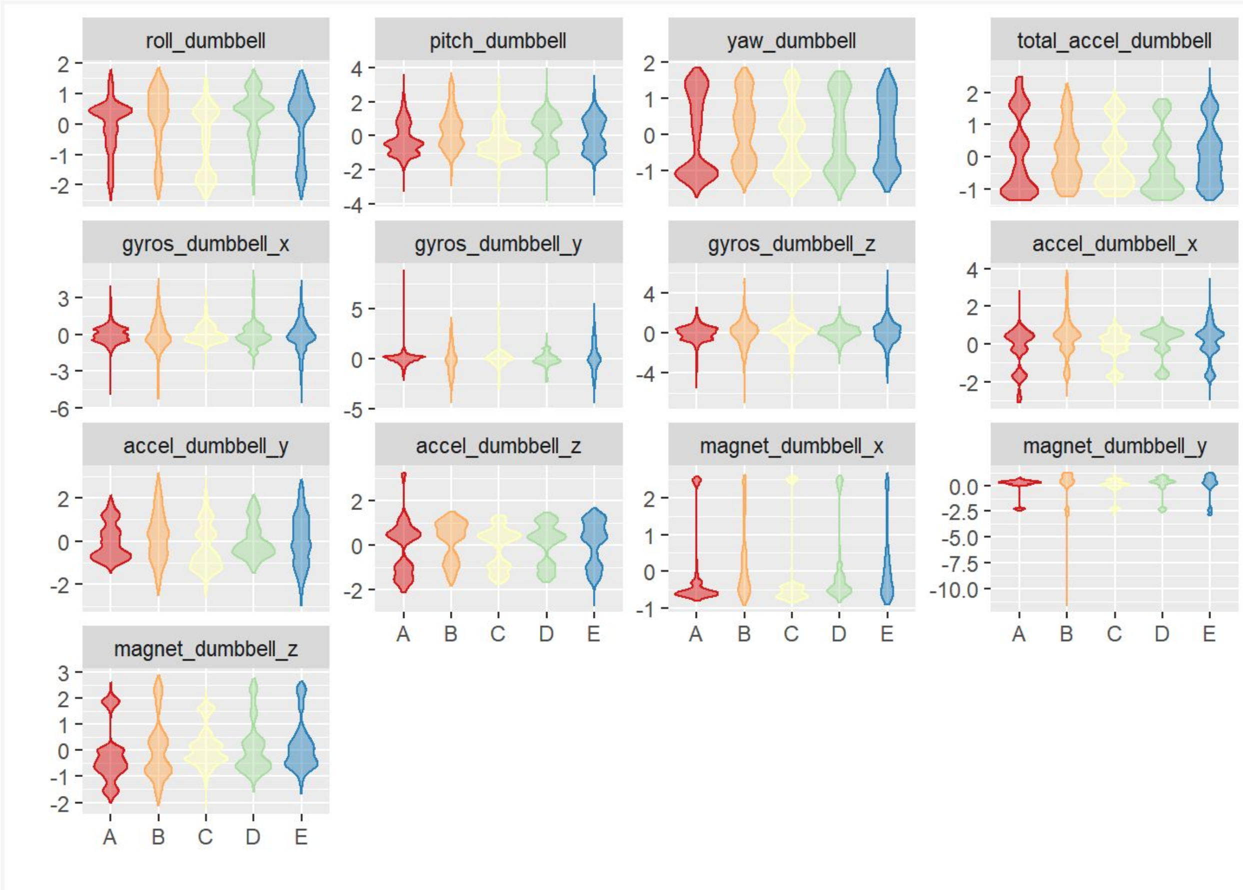
```
## Loading required package: reshape2

##
## Attaching package: 'reshape2'

## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```



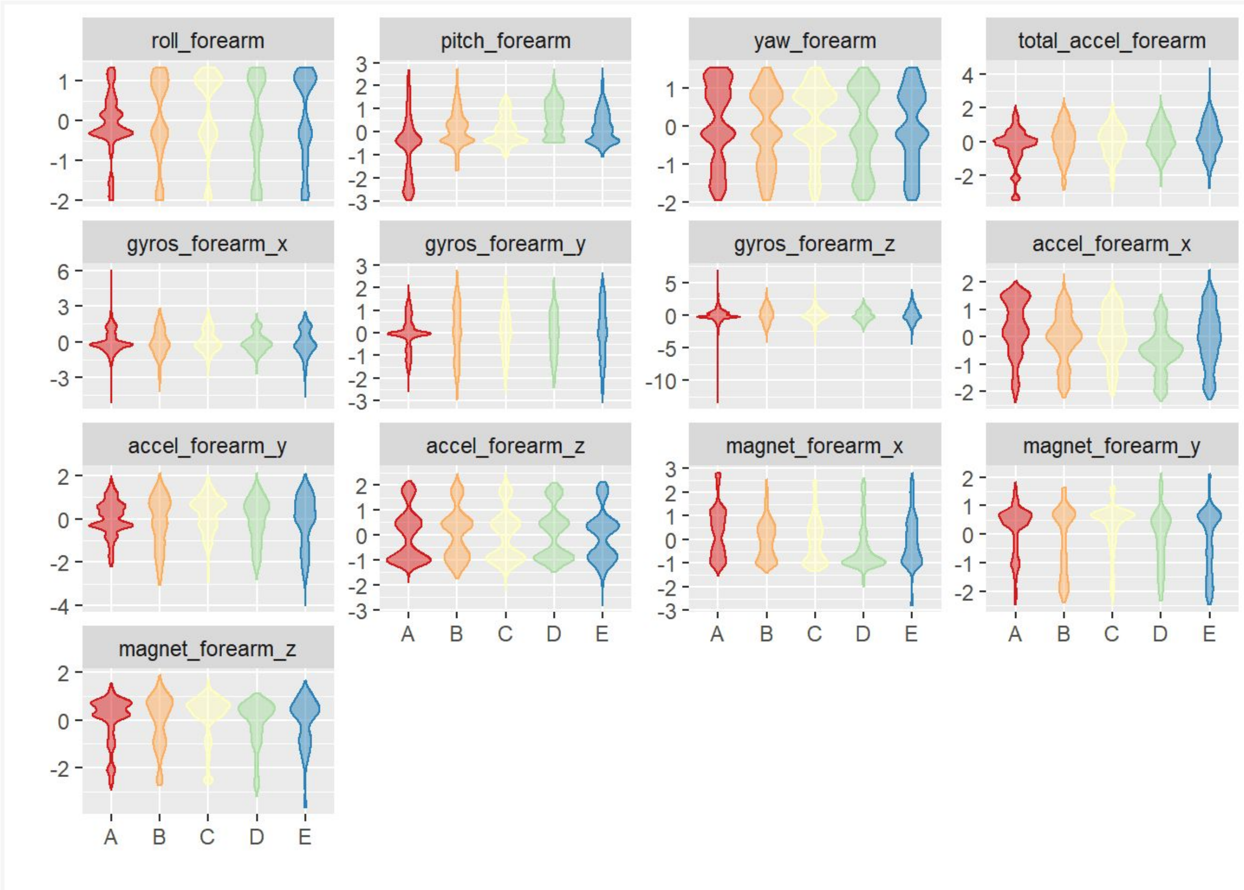

```
histGroup(DTrainCS, "[^(fore)]arm")
```




```
histGroup(DTrainCS, "dumbbell")
```




```
histGroup(DTrainCS, "forearm")
```



Prediction model

Using random forest, the out of sample error should be small. The error will be estimated using the 40% probing sample.

Setting up the parallel clusters:

```
require(parallel)
```

Loading required package: parallel

```
require(doParallel)
```

Loading required package: doParallel

Loading required package: foreach

Loading required package: iterators

```
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)
```

Setting up the control parameters:

```
ctrl <- trainControl(classProbs=TRUE,
                     savePredictions=TRUE,
                     allowParallel=TRUE)
```

Fitting the model over the tuning parameters:

```
method <- "rf"
system.time(trainingModel <- train(classe ~ ., data=DTrainCS, method=method))
```

```
##    user  system elapsed
##   55.25    1.33 2579.08
```

Stopping the clusters:

```
stopCluster(cl)
```

Model evaluation - training dataset

```
trainingModel
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9855573  0.9817181
##   27    0.9858023  0.9820276
##   52    0.9783196  0.9725532
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
hat <- predict(trainingModel, DTrainCS)
confusionMatrix(hat, DTrain[, classe])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
```

```
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Model evaluation - probing dataset

```
hat <- predict(trainingModel, DProbeCS)
confusionMatrix(hat, DProbeCS[, classe])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   15    0    0    0
##          B    0 1499    5    1    0
##          C    0    4 1362   14    2
##          D    0    0    1 1271    4
##          E    1    0    0    0 1436
##
## Overall Statistics
##
##                Accuracy : 0.994
##                  95% CI : (0.992, 0.9956)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9924
```

```
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9875   0.9956   0.9883   0.9958
## Specificity            0.9973   0.9991   0.9969   0.9992   0.9998
## Pos Pred Value         0.9933   0.9960   0.9855   0.9961   0.9993
## Neg Pred Value         0.9998   0.9970   0.9991   0.9977   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1911   0.1736   0.1620   0.1830
## Detection Prevalence   0.2863   0.1918   0.1761   0.1626   0.1832
## Balanced Accuracy      0.9984   0.9933   0.9963   0.9938   0.9978
```

Final model

```
varImp(trainingModel)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt            100.000
## pitch_forearm         60.097
## yaw_belt              54.225
## magnet_dumbbell_z     44.165
## magnet_dumbbell_y     43.297
## pitch_belt            42.046
## roll_forearm          40.575
## accel_dumbbell_y      21.208
## roll_dumbbell         19.079
## magnet_dumbbell_x     17.565
## accel_forearm_x       17.290
## magnet_belt_z         15.203
## magnet_belt_y         13.385
## total_accel_dumbbell  13.351
## magnet_forearm_z      12.728
## accel_belt_z          12.615
## accel_dumbbell_z      12.464
## gyros_belt_z          11.478
## yaw_arm               10.839
## accel_forearm_z        8.951
```

```
trainingModel$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: 27
##
##         OOB estimate of  error rate: 0.86%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3343    3    1    0    1 0.001493429
## B   24 2247    7    1    0 0.014041246
## C    0    8 2036   10    0 0.008763389
## D    0    1   27 1899    3 0.016062176
## E    0    2    5    8 2150 0.006928406
```

We see that the **estimated error rate, is less than 1%.**

Saving the training model object, for later:

```
save(trainingModel, file="trainingModel.RData")
```

Test data prediction

Loading the training model:

```
load(file="trainingModel.RData", verbose=TRUE)
```

```
## Loading objects:
##   trainingModel
```

Getting the predictions and evaluating them:

```
DTestCS <- predict(preProc, DTest[, predCandidates, with=FALSE])
hat <- predict(trainingModel, DTestCS)
DTest <- cbind(hat , DTest)
subset(DTest, select=names(DTest)[grep("belt|[^(fore)]arm|dumbbell|forearm", names(DTest),
invert=TRUE)])
```

```
##     hat V1 user_name raw_timestamp_part_1 raw_timestamp_part_2   cvtd_timestamp
##  1:   B  1     pedro           1323095002               868349 05/12/2011 14:23
##  2:   A  2    jeremy           1322673067               778725 30/11/2011 17:11
##  3:   B  3    jeremy           1322673075               342967 30/11/2011 17:11
##  4:   A  4    adelmo           1322832789               560311 02/12/2011 13:33
##  5:   A  5    eurico           1322489635               814776 28/11/2011 14:13
##  6:   E  6    jeremy           1322673149               510661 30/11/2011 17:12
##  7:   D  7    jeremy           1322673128               766645 30/11/2011 17:12
##  8:   B  8    jeremy           1322673076                54671 30/11/2011 17:11
##  9:   A  9  carlitos           1323084240               916313 05/12/2011 11:24
## 10:   A 10   charles           1322837822               384285 02/12/2011 14:57
## 11:   B 11  carlitos           1323084277                36553 05/12/2011 11:24
```

```
## 12:   C 12    jeremy          1322673101              442731 30/11/2011 17:11
## 13:   B 13    eurico          1322489661              298656 28/11/2011 14:14
## 14:   A 14    jeremy          1322673043              178652 30/11/2011 17:10
## 15:   E 15    jeremy          1322673156              550750 30/11/2011 17:12
## 16:   E 16    eurico          1322489713              706637 28/11/2011 14:15
## 17:   A 17     pedro          1323094971              920315 05/12/2011 14:22
## 18:   B 18  carlitos          1323084285              176314 05/12/2011 11:24
## 19:   B 19     pedro          1323094999              828379 05/12/2011 14:23
## 20:   B 20    eurico          1322489658              106658 28/11/2011 14:14
##     new_window num_window problem_id
##  1:         no         74          1
##  2:         no        431          2
##  3:         no        439          3
##  4:         no        194          4
##  5:         no        235          5
##  6:         no        504          6
##  7:         no        485          7
##  8:         no        440          8
##  9:         no        323          9
## 10:         no        664         10
## 11:         no        859         11
## 12:         no        461         12
## 13:         no        257         13
## 14:         no        408         14
## 15:         no        779         15
## 16:         no        302         16
## 17:         no         48         17
## 18:         no        361         18
## 19:         no         72         19
## 20:         no        255         20
```

Submitting

The submission files are saved into the folder: /Users/user/Desktop/Data Science_R/PML_files

```
pml_write_files = function(x){
  n = length(x)
  path <- "/Users/user/Desktop/Data Science_R/PML_files"
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=file.path(path, filename),quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
pml_write_files(hat)
```