Prediction Assignment Writeup

Practical Machine Learning Project

Data Science Specialization @ Coursera.org (Johns Hopkins University)

Author: Mercia Carolina Wentzel

Executive Summary

In a 2013 study named "Qualitative Activity Recognition of Weight Lifting Exercises" by Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H, the authors investigated whether quality rather than quantity of exercise could be assessed from devices such as Jawbone Up, Nike Fuelband and Fitbit. During the study, data was collected from accellerometers on the belt, forearm, arm, and dumbell of 6 volunteers who were asked to perform barbell lifts both correctly and in different incorrect ways. Information about the study, including a link to the weight lifting exercises (WLE) dataset, were published here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

For this project, training and test datasets originating from the published dataset mentioned above were used to (a) build a model that can predict the way in which barbell lifts are performed, and (b) test the model.

```
## Warning: package 'caret' was built under R version 3.5.1

## Warning: package 'ggplot2' was built under R version 3.5.1

## Warning: package 'rattle' was built under R version 3.5.1

## Warning: package 'knitr' was built under R version 3.5.1
```

Get and clean training data

Read training data and remove columns that contain no data as well as static data columns:

```
train_orig <- read.csv("pml-training.csv")
dim(train_orig)

## [1] 19622   160

count_nodata <- sapply(train_orig, function(y) length(which(is.na(y))) + leng
th(which(y=="")))
unique(count_nodata)
```

```
## [1]     0 19216

train_bool <- sapply(
    train_orig, function(y) as.logical(length(which(is.na(y))) + length(which
(y=="")) > 0))
train_clean <- train_orig[, -which(train_bool)]
colnames(train_clean)

##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
##  [7] "num_window"           "roll_belt"            "pitch_belt"
## [10] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
## [13] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [16] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [19] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [22] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [25] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [28] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [31] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [34] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [37] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [40] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [43] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [46] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [49] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [52] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [55] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [58] "magnet_forearm_y"     "magnet_forearm_z"     "classe"

head(rbind(head(train_clean[,1:7], 3), tail(train_clean[,1:7], 3)))

##           X user_name raw_timestamp_part_1 raw_timestamp_part_2
## 1         1  carlitos           1323084231               788290
## 2         2  carlitos           1323084231               808298
## 3         3  carlitos           1323084231               820366
## 19620 19620    adelmo           1322832937               636283
## 19621 19621    adelmo           1322832937               964299
## 19622 19622    adelmo           1322832937               972293
##         cvtd_timestamp new_window num_window
## 1     05/12/2011 11:23         no         11
## 2     05/12/2011 11:23         no         11
## 3     05/12/2011 11:23         no         11
## 19620 02/12/2011 13:35         no        864
## 19621 02/12/2011 13:35         no        864
## 19622 02/12/2011 13:35        yes        864

train_clean <- train_clean[, -c(1:7)]
dim(train_clean)

## [1] 19622    53
```

```
count_nodata <- sapply(train_clean, function(y) length(which(is.na(y))) + len
gth(which(y=="")))
unique(count_nodata)

## [1] 0
```

Get and clean test data

Read test data and remove same columns as those that were removed from training dataset:

```
test_orig <- read.csv("pml-testing.csv")
dim(test_orig)

## [1]  20 160

test_clean <- test_orig[, -which(train_bool)]
test_clean <- test_clean[, -c(1:7)]
dim(test_clean)

## [1] 20 53

count_nodata <- sapply(test_clean, function(y) length(which(is.na(y))) + leng
th(which(y=="")))
unique(count_nodata)

## [1] 0
```

Partition training data

Split the training dataset into training and validation partitions:

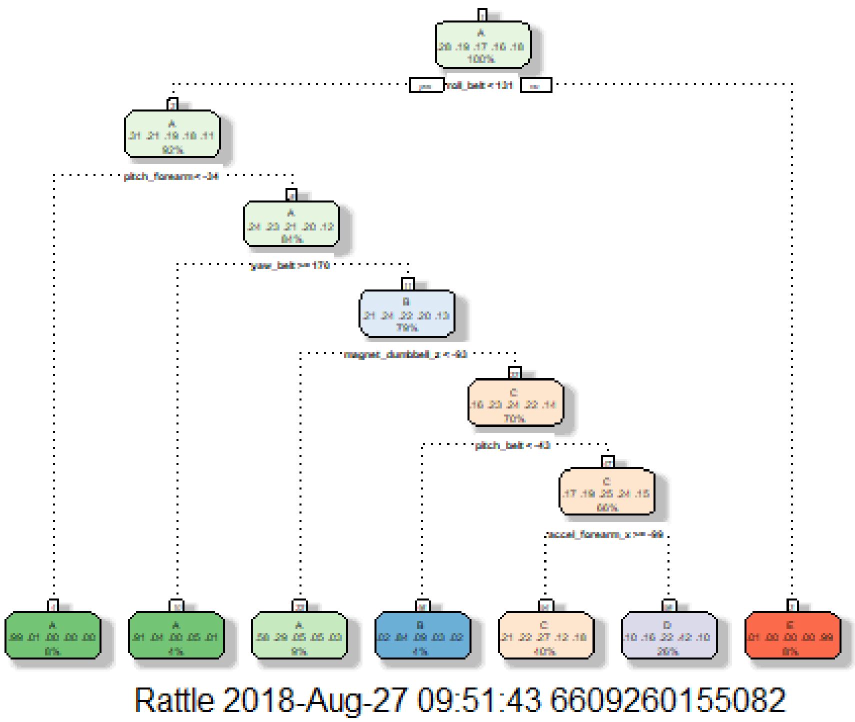
```
set.seed(12345)
train_clean_partition <- createDataPartition(train_clean$classe, p = 0.6, lis
t = FALSE)
train_in <- train_clean[train_clean_partition, ]
train_out <- train_clean[-train_clean_partition, ]
```

Classification and regression tree (CART) model

Fit a CART model on the training partition:

```
start_CART <- Sys.time()
```

```
set.seed(12345)
ctl <- trainControl(method = "cv", number = 5) ## 5 resamplings using "cv" me
thod
train_in_CART <- train(classe ~ ., data = train_in, method = "rpart", trContr
ol = ctl)
fancyRpartPlot(train_in_CART$finalModel)
```



Cross-validate the CART model that has been fitted on the training partition against the validation partition:

```
predict_CART <- predict(train_in_CART, train_out)
matrix_CART <- confusionMatrix(train_out$classe, predict_CART)
matrix_CART

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1357    3  693  171    8
##          B  229  259  725  305    0
##          C   38   28  819  483    0
##          D   66    8  389  823    0
##          E   15   10  557  200  660
##
## Overall Statistics
##
##                Accuracy : 0.4994
##                  95% CI : (0.4882, 0.5105)
```

```
##     No Information Rate : 0.4057
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3764
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7959  0.84091   0.2573   0.4152  0.98802
## Specificity            0.8575  0.83298   0.8823   0.9210  0.89106
## Pos Pred Value         0.6080  0.17062   0.5987   0.6400  0.45770
## Neg Pred Value         0.9380  0.99226   0.6351   0.8233  0.99875
## Prevalence             0.2173  0.03926   0.4057   0.2526  0.08514
## Detection Rate         0.1730  0.03301   0.1044   0.1049  0.08412
## Detection Prevalence   0.2845  0.19347   0.1744   0.1639  0.18379
## Balanced Accuracy      0.8267  0.83694   0.5698   0.6681  0.93954
```

```
difftime_CART <- round(difftime(Sys.time(), start_CART, units = "mins"), 0)
```

Proceed to see if model accuracy can be improved:

Generalized Boosted Model (GBM)

Fit a GBM on the training partition:

```
start_GBM <- Sys.time()

set.seed(12345)
ctl <- trainControl(method = "repeatedcv", number = 5,  ## 5 resamplings usin
g "repeatcv" method
                    repeats = 1)
train_in_GBM <- train(classe ~ ., data = train_in, method = "gbm", trControl
= ctl,
                      verbose = FALSE)
train_in_GBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 44 had non-zero influence.
```

Cross-validate the GBM that has been fitted on the training partition against the validation parition:

```
predict_GBM <- predict(train_in_GBM, train_out)
matrix_GBM <- confusionMatrix(train_out$classe, predict_GBM)
matrix_GBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2203   22    3    4    0
##          B   45 1429   37    5    2
##          C    0   37 1312   18    1
##          D    1    5   51 1221    8
##          E    0   25   10   20 1387
##
## Overall Statistics
##
##                Accuracy : 0.9625
##                  95% CI : (0.9581, 0.9666)
##     No Information Rate : 0.2866
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9526
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9795   0.9414   0.9285   0.9629   0.9921
## Specificity            0.9948   0.9859   0.9913   0.9901   0.9915
## Pos Pred Value         0.9870   0.9414   0.9591   0.9495   0.9619
## Neg Pred Value         0.9918   0.9859   0.9844   0.9928   0.9983
## Prevalence             0.2866   0.1935   0.1801   0.1616   0.1782
## Detection Rate         0.2808   0.1821   0.1672   0.1556   0.1768
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9872   0.9637   0.9599   0.9765   0.9918

difftime_GBM <- round(difftime(Sys.time(), start_GBM, units = "mins"), 0)
```

Model accuracy has been improved significantly. Proceed to see if it can be improved further:

Random Forest (RF) model

Fit a RF model on the training partition:

```
start_RF <- Sys.time()

set.seed(12345)
ctl <- trainControl(method = "cv", number = 5) ## 5 resamplings using "cv" me
thod
train_in_RF <- train(classe ~ ., data = train_in, method = "rf", trControl =
```

```
ctl)
train_in_RF$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: 2
##
##         OOB estimate of  error rate: 0.81%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3344    4    0    0    0 0.001194743
## B   16 2254    8    0    1 0.010969724
## C    1   17 2033    3    0 0.010223953
## D    0    0   35 1892    3 0.019689119
## E    0    0    1    6 2158 0.003233256
```

Cross-validate the RF model that has been fitted on the training partition against the validation parition:

```
predict_RF <- predict(train_in_RF, train_out)
matrix_RF <- confusionMatrix(train_out$classe, predict_RF)
matrix_RF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    3    0    0    0
##          B   11 1503    4    0    0
##          C    0   10 1357    1    0
##          D    0    0   24 1259    3
##          E    0    0    1    4 1437
##
## Overall Statistics
##
##                Accuracy : 0.9922
##                  95% CI : (0.99, 0.994)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9902
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9951   0.9914   0.9791   0.9960   0.9979
## Specificity            0.9995   0.9976   0.9983   0.9959   0.9992
```

```
## Pos Pred Value         0.9987   0.9901   0.9920   0.9790   0.9965
## Neg Pred Value         0.9980   0.9979   0.9955   0.9992   0.9995
## Prevalence             0.2855   0.1932   0.1767   0.1611   0.1835
## Detection Rate         0.2841   0.1916   0.1730   0.1605   0.1832
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9973   0.9945   0.9887   0.9960   0.9986

difftime_RF <- round(difftime(Sys.time(), start_RF, units = "mins"), 0)
```

Model accuracy has been improved even further. Proceed to tabulate results and select a model:

Comparison and selection

Table:

Metric	CART	GBM	RF
Accuracy	49.94%	96.25%	99.22%
Out of sample error	50.06%	3.75%	0.78%
No Information Rate	40.57%	28.66%	28.55%
Time (minutes)	0	3	6

Comparison:

The classification and regression tree (CART) model has performed quite poorly in terms of accuracy, so it has not been considered any further. Both the generalized boosted model (GBM) and the random forest (RF) model achieved over 96% accuracy, with RF achieving near perfection at more than 99%.

Here, the out of sample errors have been estimated as the respective reciprocates of the accuracy percentages.

Both "no information" rates were between 28% and 29%, which I interpreted as low enough to warrant acceptance of the accuracy percentages.

Notably, the execution time of GBM is about twice as fast as the execution time of RF. This is by no means a proper CPU performance test, but it does give a comparitive indication of execution times.

Selection:

Since execution time has not been an issue for this project, the highly accurate RF model has been selected to predict the way in which barbell lifts were performed as per the test dataset.

Prediction

Preamble:

As specified by the study authors, the five ways in which exercises can be performed are:
* correctly (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).

Predict ways in which 20 respective exercises in the test dataset were performed:

```
predict(train_in_RF, test_clean)

##  [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
```