Practical Machine Learning Course Project

Predicting the manner people excercise

Jorge E. Ibarra-Esquer

2025-01-15

Abstract

This project presents an experiment on prediction of the manner which people exercise. Data is obtained from commercial gadgets containing accelerometers. A random forest model is used for prediction, achieving an accuracy of 0.99.

Introduction

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, our 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.

Training data are available from:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

Test data are available from:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Information about the data was not available on the provided website. A Google search was performed and the following description was found at https://rpubs.com/chirozie/656543 (https://rpubs.com/chirozie/656543)

Data description

The outcome variable is “*classe*”, a factor variable with 5 levels. For this data set, participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 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)
- throwing the hips to the front (Class E)

Data exploration

We start by loading the datasets and storing them into local objects.

```r
library(readr)
library(caret)
library(randomForest)
training <- read_csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
testing <-read_csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
```

We observe that the training dataset contains 19622 observations, while the testing dataset contains a total of 20 observations. There are 160 variables in the datasets, named as follows:

```r
names(training)
```

```
##   [1] "...1"                     "user_name"
##   [3] "raw_timestamp_part_1"     "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"           "new_window"
##   [7] "num_window"               "roll_belt"
##   [9] "pitch_belt"               "yaw_belt"
##  [11] "total_accel_belt"         "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"      "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"       "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"        "max_roll_belt"
##  [19] "max_picth_belt"           "max_yaw_belt"
##  [21] "min_roll_belt"            "min_pitch_belt"
##  [23] "min_yaw_belt"             "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"     "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"     "avg_roll_belt"
##  [29] "stddev_roll_belt"         "var_roll_belt"
##  [31] "avg_pitch_belt"           "stddev_pitch_belt"
##  [33] "var_pitch_belt"           "avg_yaw_belt"
##  [35] "stddev_yaw_belt"          "var_yaw_belt"
##  [37] "gyros_belt_x"             "gyros_belt_y"
##  [39] "gyros_belt_z"             "accel_belt_x"
##  [41] "accel_belt_y"             "accel_belt_z"
##  [43] "magnet_belt_x"            "magnet_belt_y"
##  [45] "magnet_belt_z"            "roll_arm"
##  [47] "pitch_arm"                "yaw_arm"
##  [49] "total_accel_arm"          "var_accel_arm"
##  [51] "avg_roll_arm"             "stddev_roll_arm"
##  [53] "var_roll_arm"             "avg_pitch_arm"
##  [55] "stddev_pitch_arm"         "var_pitch_arm"
##  [57] "avg_yaw_arm"              "stddev_yaw_arm"
##  [59] "var_yaw_arm"              "gyros_arm_x"
##  [61] "gyros_arm_y"              "gyros_arm_z"
##  [63] "accel_arm_x"              "accel_arm_y"
##  [65] "accel_arm_z"              "magnet_arm_x"
##  [67] "magnet_arm_y"             "magnet_arm_z"
##  [69] "kurtosis_roll_arm"        "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"         "skewness_roll_arm"
##  [73] "skewness_pitch_arm"       "skewness_yaw_arm"
##  [75] "max_roll_arm"             "max_picth_arm"
##  [77] "max_yaw_arm"              "min_roll_arm"
##  [79] "min_pitch_arm"            "min_yaw_arm"
##  [81] "amplitude_roll_arm"       "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"        "roll_dumbbell"
##  [85] "pitch_dumbbell"           "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"   "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"    "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell"  "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"        "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"         "min_roll_dumbbell"
##  [97] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
## [101] "amplitude_yaw_dumbbell"   "total_accel_dumbbell"
## [103] "var_accel_dumbbell"       "avg_roll_dumbbell"
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "classe"
```

Data preprocessing

Early exploration of data shows that several variables do not provide information useful for the purposes of this experiment. The following variables will be removed from both testing and training datasets:

```
t(t(head(names(training),7)))
```

```
##      [,1]
## [1,] "...1"
## [2,] "user_name"
## [3,] "raw_timestamp_part_1"
## [4,] "raw_timestamp_part_2"
## [5,] "cvtd_timestamp"
## [6,] "new_window"
## [7,] "num_window"
```

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

In addition, several other variables contain mostly NA observations. As they do not provide useful information, columns with more that 85% NA values will be removed.

```
thr<-dim(training)[1]*0.85

colRemove<-colnames(training)[colSums(is.na(training))>thr]

colRemove
```

```
##   [1] "kurtosis_roll_belt"       "kurtosis_picth_belt"
##   [3] "kurtosis_yaw_belt"        "skewness_roll_belt"
##   [5] "skewness_roll_belt.1"     "skewness_yaw_belt"
##   [7] "max_roll_belt"            "max_picth_belt"
##   [9] "max_yaw_belt"             "min_roll_belt"
##  [11] "min_pitch_belt"           "min_yaw_belt"
##  [13] "amplitude_roll_belt"      "amplitude_pitch_belt"
##  [15] "amplitude_yaw_belt"       "var_total_accel_belt"
##  [17] "avg_roll_belt"            "stddev_roll_belt"
##  [19] "var_roll_belt"            "avg_pitch_belt"
##  [21] "stddev_pitch_belt"        "var_pitch_belt"
##  [23] "avg_yaw_belt"             "stddev_yaw_belt"
##  [25] "var_yaw_belt"             "var_accel_arm"
##  [27] "avg_roll_arm"             "stddev_roll_arm"
##  [29] "var_roll_arm"             "avg_pitch_arm"
##  [31] "stddev_pitch_arm"         "var_pitch_arm"
##  [33] "avg_yaw_arm"              "stddev_yaw_arm"
##  [35] "var_yaw_arm"              "kurtosis_roll_arm"
##  [37] "kurtosis_picth_arm"       "kurtosis_yaw_arm"
##  [39] "skewness_roll_arm"        "skewness_pitch_arm"
##  [41] "skewness_yaw_arm"         "max_roll_arm"
##  [43] "max_picth_arm"            "max_yaw_arm"
##  [45] "min_roll_arm"             "min_pitch_arm"
##  [47] "min_yaw_arm"              "amplitude_roll_arm"
##  [49] "amplitude_pitch_arm"      "amplitude_yaw_arm"
##  [51] "kurtosis_roll_dumbbell"   "kurtosis_picth_dumbbell"
##  [53] "kurtosis_yaw_dumbbell"    "skewness_roll_dumbbell"
##  [55] "skewness_pitch_dumbbell"  "skewness_yaw_dumbbell"
##  [57] "max_roll_dumbbell"        "max_picth_dumbbell"
##  [59] "max_yaw_dumbbell"         "min_roll_dumbbell"
##  [61] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [63] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
##  [65] "amplitude_yaw_dumbbell"   "var_accel_dumbbell"
##  [67] "avg_roll_dumbbell"        "stddev_roll_dumbbell"
##  [69] "var_roll_dumbbell"        "avg_pitch_dumbbell"
##  [71] "stddev_pitch_dumbbell"    "var_pitch_dumbbell"
##  [73] "avg_yaw_dumbbell"         "stddev_yaw_dumbbell"
##  [75] "var_yaw_dumbbell"         "kurtosis_roll_forearm"
##  [77] "kurtosis_picth_forearm"   "kurtosis_yaw_forearm"
##  [79] "skewness_roll_forearm"    "skewness_pitch_forearm"
##  [81] "skewness_yaw_forearm"     "max_roll_forearm"
##  [83] "max_picth_forearm"        "max_yaw_forearm"
##  [85] "min_roll_forearm"         "min_pitch_forearm"
##  [87] "min_yaw_forearm"          "amplitude_roll_forearm"
##  [89] "amplitude_pitch_forearm"  "amplitude_yaw_forearm"
##  [91] "var_accel_forearm"        "avg_roll_forearm"
##  [93] "stddev_roll_forearm"      "var_roll_forearm"
##  [95] "avg_pitch_forearm"        "stddev_pitch_forearm"
##  [97] "var_pitch_forearm"        "avg_yaw_forearm"
##  [99] "stddev_yaw_forearm"       "var_yaw_forearm"
```

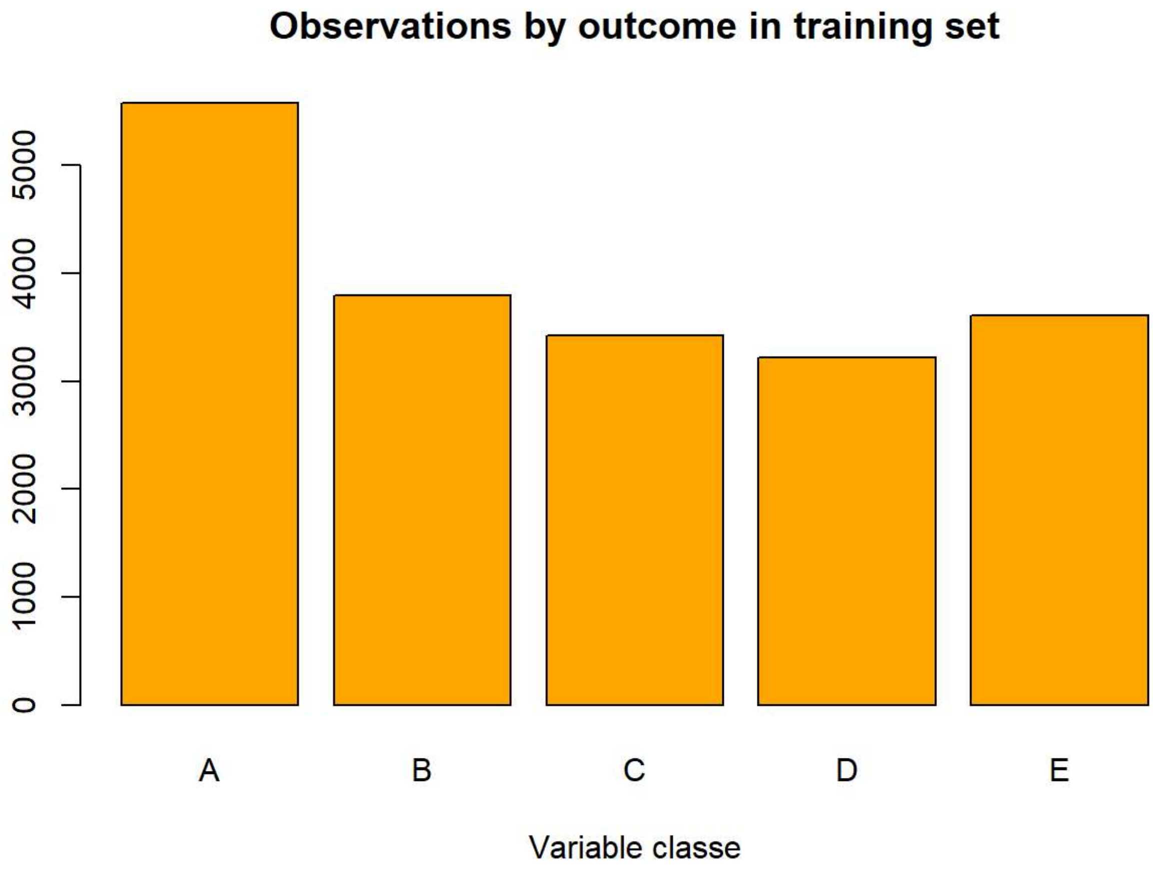
```
colArray=colSums(is.na(trainingClean))>thr
trainingClean<-trainingClean[,!colArray]
testingClean<-testingClean[,!colArray]
```

After this step, 53 variables remain in each dataset.

Analysis of cleaned dataset

Outcome is expressed as 5 levels in variable classe. The following graph shows the distribution of these levels in the training dataset.

```
barplot(table(trainingClean$classe), col="orange",
        xlab="Variable classe",
        main="Observations by outcome in training set")
```



The plot shows that the most common case is A, indicating that the exercise is performed exactly as specified. The rest of the levels are very close to each other in number of occurrences.

Training a prediction model

A Random Forest model will be trained to predict the levels in the testing dataset. First, the training dataset will be split as 75% for training and 25% for validation.

Partitioning the dataset

```
set.seed(24680)

forTrain <- createDataPartition(trainingClean$classe, p = 0.75, list = FALSE)
validSet <- trainingClean[-forTrain, ]
trainSet <- trainingClean[forTrain, ]
```

The training dataset consists of 14718 observations. The validation dataset consists of 4904 observations.

Random forest prediction model

A random forest model is fit for prediction using a 10-fold cross validation.

```
rForest <- train(classe ~ ., data = trainSet, method = "rf", trControl = trainControl(method = "cv", 10), n
tree = 200)
rForest
```

```
## Random Forest 
## 
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E' 
## 
## No pre-processing
## Resampling: Cross-Validated (10 fold) 
## Summary of sample sizes: 13248, 13246, 13245, 13246, 13245, 13248, ... 
## Resampling results across tuning parameters:
## 
##   mtry  Accuracy  Kappa    
##    2    0.992526  0.9905447
##   27    0.991440  0.9891710
##   52    0.985393  0.9815215
## 
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

The trained model has an accuracy of 0.992526. Now it will be validated using the validation dataset.

```
valPrediction <- predict(rForest, validSet)

confValid <- confusionMatrix(valPrediction, factor(validSet$classe))

valValues=confValid$overall
names(valValues)<-NULL
```

Accuracy of the model with the validation dataset is 0.9932708 and an out of sample error of 0.6729201%. The confusion matrix shows the results of validation.

```
confValid
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    5    0    0    0
##          B    0  943    4    0    0
##          C    0    1  851   21    0
##          D    0    0    0  783    2
##          E    0    0    0    0  899
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9933          
##                  95% CI : (0.9906, 0.9954)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9915          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9937   0.9953   0.9739   0.9978
## Specificity            0.9986   0.9990   0.9946   0.9995   1.0000
## Pos Pred Value         0.9964   0.9958   0.9748   0.9975   1.0000
## Neg Pred Value         1.0000   0.9985   0.9990   0.9949   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1923   0.1735   0.1597   0.1833
## Detection Prevalence   0.2855   0.1931   0.1780   0.1601   0.1833
## Balanced Accuracy      0.9993   0.9963   0.9949   0.9867   0.9989
```

Prediction of the manner in which people exercise

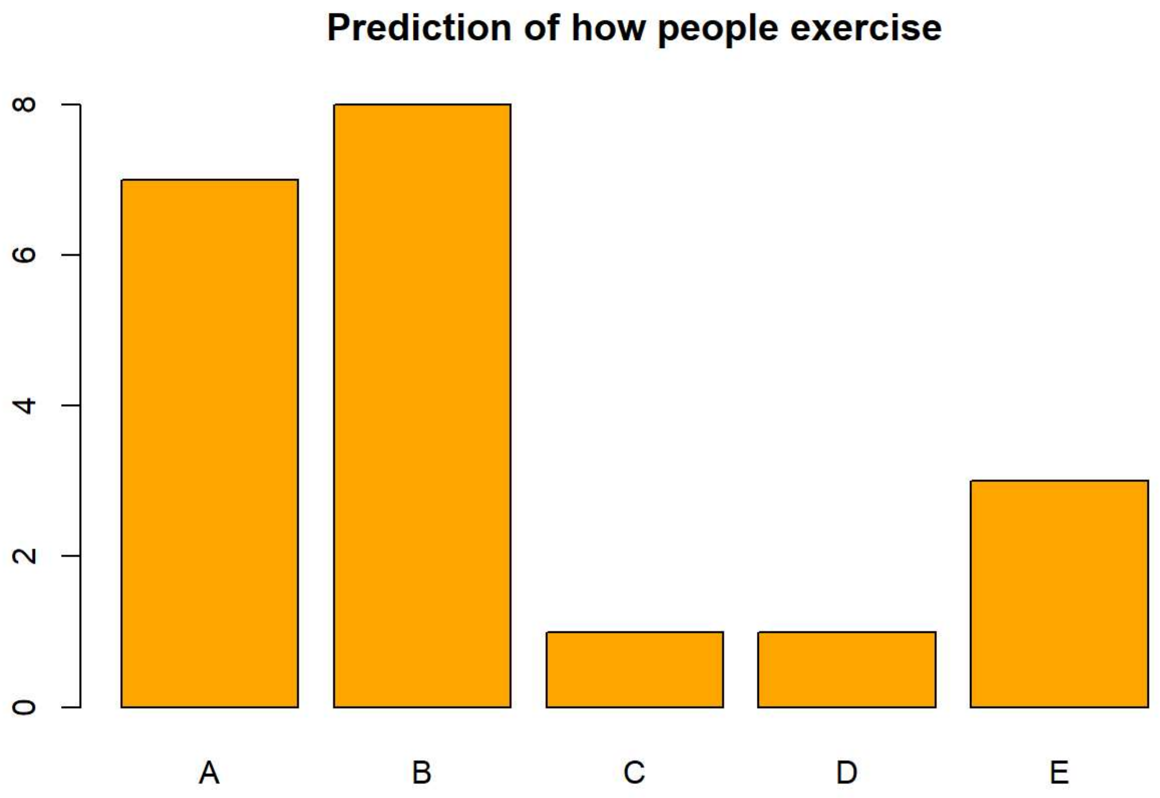
Finally, the prediction will be performed on the testing dataset.

```
testPrediction <- predict(rForest, testingClean)

testPrediction
```

```
##  [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
```

```
barplot(table(testPrediction), col="orange",
        main="Prediction of how people exercise")
```



Conclusion

A random forest model was trained for prediction of the manner people exercise. The trained model has an accuracy of 0.992526 over a 10-fold cross validation.

Prediction using a validation dataset had an accuracy of 0.9932708. A final test was performed on a 20-sample dataset.