Predicting Correctness of Weight Lifting Exercises

Paolo Saracco

2024-06-09

Summary

We analyze the Weight Lifting Exercises Dataset, which gathers data from accelerometers on the belt, forearm, arm, and dumbell of 6 people performing barbell lifts correctly and incorrectly in 5 different ways. After fitting a predictive model for activity recognition based on these data, we use it to predict the correctness of the exercises performed by new users.

Packages

We will use the `dplyr` package for working with data frames, the `ggplot2` package for graphs, the `caret` package for training and predicting.

```
library(dplyr); library(ggplot2); library(caret); set.seed(0)
```

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 webpage of the project (see the section on the Weight Lifting Exercise Dataset).

Data

The data set originally comes from the Weight Lifting Exercises Dataset (see the section on the Weight Lifting Exercise Dataset on the webpage of the project - possibly see the web archive version of it at web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har). The training data for this project are also available at:

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

The test data are also available at:

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

We downloaded the data into a `data` folder as `train.csv` and `test.csv`.

```
raw_training <- read.csv("data/train.csv")
raw_testing <- read.csv("data/test.csv")
```

Exploratory data analysis

We are dealing with a 19622 x 160 training data set and a 20 x 160 test data set. Let us inspect the training data and leave aside the test data set. Details are suppressed for the sake of brevity, but we explicitly show the names of the variables in the data set for the convenience of the reader.

```
head(raw_training)
tail(raw_training)
str(raw_training)
summary(raw_training)
```

```
names(raw_training)
```

```
##   [1] "X"                        "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

The first 7 columns contain data that are not useful for our purposes, hence we drop them. We also convert to numeric those columns that are erroneously interpreted as integers or characters. Then we drop those columns that are mostly (at least 90%) NAs. Finally, we check whether there are columns with almost zero variance, but there are not and we consider ourselves satisfied.

```
training <- raw_training %>%
        select(!c(X, user_name,
                  raw_timestamp_part_1,
                  raw_timestamp_part_2,
                  cvtd_timestamp,
                  new_window,
                  num_window)) %>%
        mutate(across(!classe, as.numeric),
               classe = factor(classe)) %>%
        select(where(function(X) { mean(is.na(X)) < 0.9 }))
nzv <- nearZeroVar(training)
nzv
```

```
## integer(0)
```

```
# training <- select(training, !all_of(nzv))
dim(training)
```

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

Model selection

In order to select a model, we compare the accuracy of three models seen in the course: **decision tree**, **random forest** and **gradient-boosting trees** model.

Holdout Cross Validation

We split our training data set into a data set for training the algorithms (70% of the data) and a data set for validating the results and compare their accuracy (30% of the data).

```
inTrain <- createDataPartition(y = training$classe,
                               p = 0.7,
                               list = FALSE)
trainSet <- training[inTrain,]
validationSet <- training[-inTrain,]
```

We also decide to perform **3-fold cross validation** in the training phase, to reduce the bias and improve the prediction capability of our models.

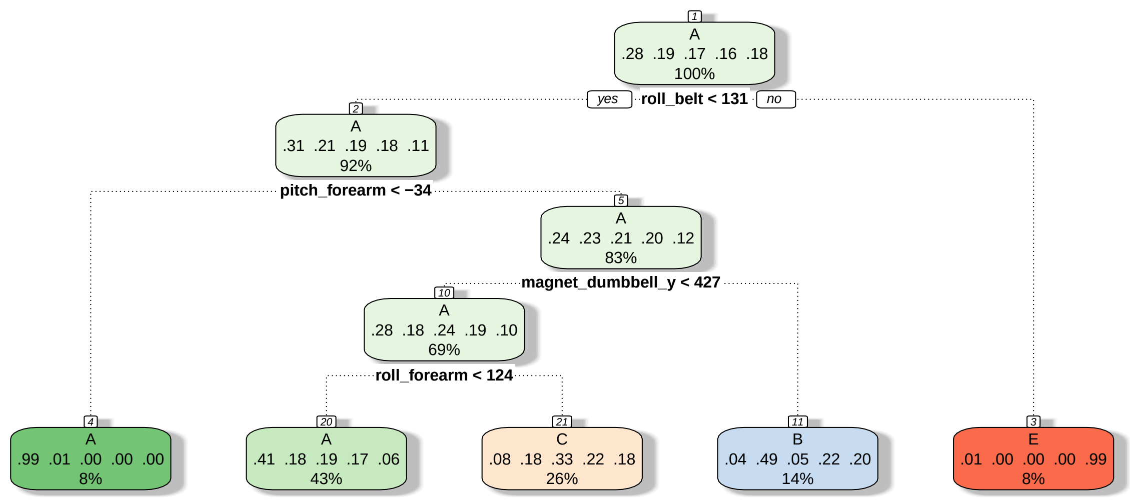
```
control <- trainControl(method = "cv", number = 3)
```

Decision tree

We start by training a single decision tree.

Creating the model Train the decision tree

```
Tree <- train(classe ~ ., data = trainSet,
              method = "rpart", trControl = control)
rattle::fancyRpartPlot(Tree$finalModel)
```



Rattle 2024−Jun−09 14:51:46 paolo

Estimation of the in sample accuracy/error Estimate accuracy on the training data

```
confusionMatrix(predict(Tree, trainSet), trainSet$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.4973429      0.3438398      0.4889467      0.5057403      0.2843416
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Check performance on validation data Predict on the validation data set and check out-of-sample performance

```
predTree <- predict(Tree, validationSet)
CnfMtxTree <- confusionMatrix(predTree, validationSet$classe)
CnfMtxTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1519  484  475  422  171
##          B   42  394   32  184  147
##          C  109  261  519  358  278
##          D    0    0    0    0    0
##          E    4    0    0    0  486
##
## Overall Statistics
##
##                Accuracy : 0.4958
##                  95% CI : (0.483, 0.5087)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3408
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                        Class: A Class: B Class: C Class: D Class: E
## Sensitivity              0.9074  0.34592  0.50585   0.0000  0.44917
## Specificity              0.6314  0.91466  0.79296   1.0000  0.99917
## Pos Pred Value           0.4946  0.49312  0.34033      NaN  0.99184
## Neg Pred Value           0.9449  0.85352  0.88372   0.8362  0.88953
## Prevalence               0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate           0.2581  0.06695  0.08819   0.0000  0.08258
## Detection Prevalence     0.5218  0.13577  0.25913   0.0000  0.08326
## Balanced Accuracy        0.7694  0.63029  0.64940   0.5000  0.72417
```

We conclude that this single decision tree has an accuracy of 0.4958369

Random forest

As second option, we train a random forest.

Creating the model Train the random forest

```
Forest <- train(classe ~ ., data = trainSet,
               method = "rf", trControl = control)
```

Estimation of the in sample accuracy/error Estimate accuracy on the training data

```
confusionMatrix(predict(Forest, trainSet), trainSet$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9997315      1.0000000      0.2843416
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Check performance on validation data Predict on the validation data set and check out-of-sample performance

```
predRF <- predict(Forest, validationSet)
CnfMtxRF <- confusionMatrix(predRF, validationSet$classe)
CnfMtxRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    5    0    0    2
##          B    2 1129    8    0    0
##          C    1    5 1012   15    0
##          D    0    0    6  948    4
##          E    0    0    0    1 1076
##
## Overall Statistics
##
##                Accuracy : 0.9917
##                  95% CI : (0.989, 0.9938)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9895
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                        Class: A Class: B Class: C Class: D Class: E
## Sensitivity              0.9982   0.9912   0.9864   0.9834   0.9945
## Specificity              0.9983   0.9979   0.9957   0.9980   0.9998
## Pos Pred Value           0.9958   0.9912   0.9797   0.9896   0.9991
## Neg Pred Value           0.9993   0.9979   0.9971   0.9968   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1918   0.1720   0.1611   0.1828
## Detection Prevalence   0.2851   0.1935   0.1755   0.1628   0.1830
## Balanced Accuracy      0.9983   0.9946   0.9910   0.9907   0.9971
```

We conclude that this random forest model has an accuracy of 0.9916737

Gradient-boosted Trees Model

Finally, we perform gradient boosting with decision trees as base learners.

Creating the model

Train the GBM

```r
GradBoostTrees <- train(classe ~ ., data = trainSet, method = "gbm",
                        verbose = FALSE, trControl = control)
```

Estimation of the in sample accuracy/error

Estimate accuracy on the training data

```r
confusionMatrix(predict(GradBoostTrees, trainSet), trainSet$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9743030      0.9674905      0.9715180      0.9768845      0.2843416
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Check performance on validation data

Predict on the validation data set and check out-of-sample performance

```r
predGBM <- predict(GradBoostTrees, validationSet)
CnfMtxGBM <- confusionMatrix(predGBM, validationSet$classe)
CnfMtxGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1645   37    0    0    4
##          B   19 1070   35    4   14
##          C    8   30  980   31    7
##          D    2    2    8  919   14
##          E    0    0    3   10 1043
##
## Overall Statistics
##
##                Accuracy : 0.9613
##                  95% CI : (0.956, 0.966)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.951
##
##  Mcnemar's Test P-Value : 2.011e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9827   0.9394   0.9552   0.9533   0.9640
## Specificity            0.9903   0.9848   0.9844   0.9947   0.9973
## Pos Pred Value         0.9757   0.9370   0.9280   0.9725   0.9877
## Neg Pred Value         0.9931   0.9855   0.9905   0.9909   0.9919
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2795   0.1818   0.1665   0.1562   0.1772
## Detection Prevalence   0.2865   0.1941   0.1794   0.1606   0.1794
## Balanced Accuracy      0.9865   0.9621   0.9698   0.9740   0.9806
```

We conclude that this gradient-boosted trees model has an accuracy of 0.9612574

Upsum and conclusion of the model selection process

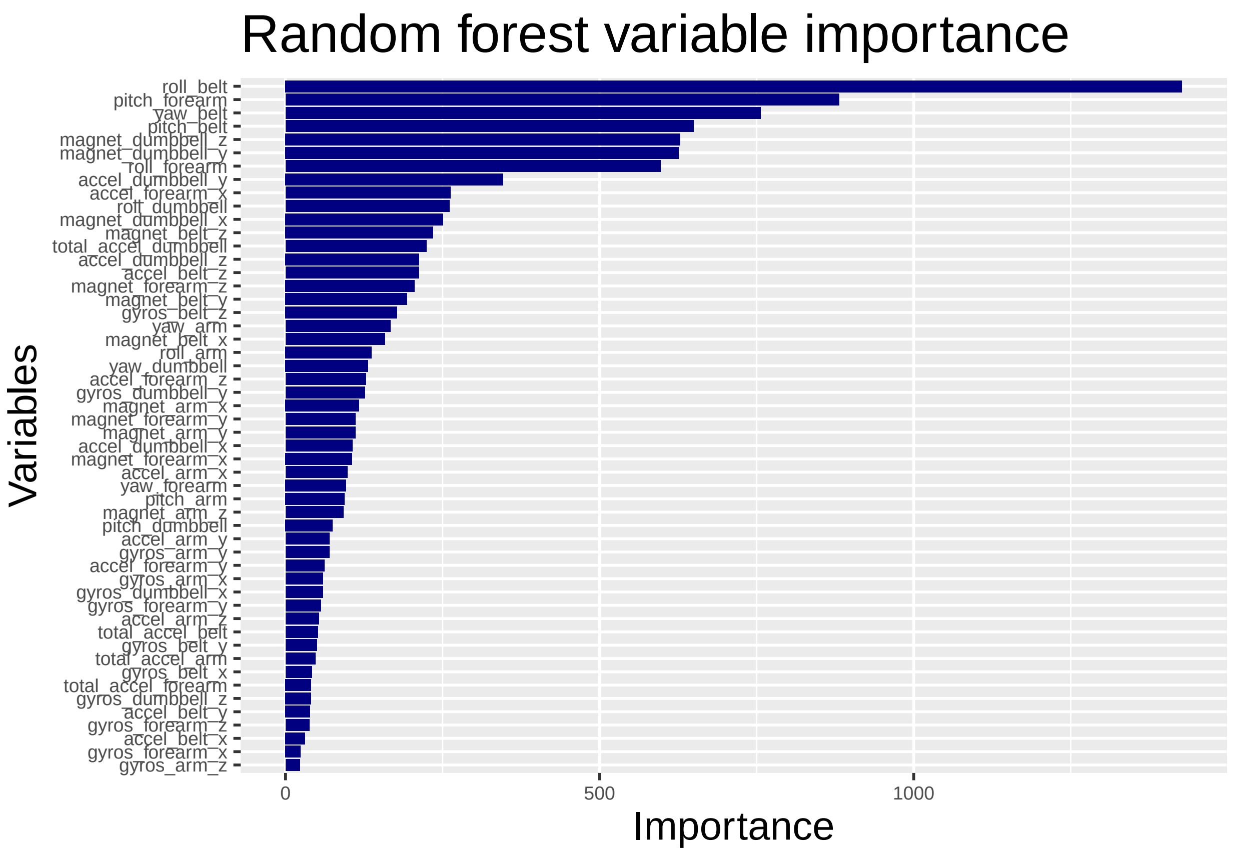
Model	In sample accuracy	In sample error	Out of sample accuracy	Out of sample error	95% CI OS Acc
Decision Tree	0.4973	0.5027	0.4958	0.5042	[0.483,0.5087]
Random Forest	1	0	0.9917	0.0083	[0.989,0.9938]
Gradient-boosted Trees	0.9743	0.0257	0.9613	0.0387	[0.956,0.966]

Therefore we decide to select the random forest model, since it performed pretty well on the testing data having a 99.17% out-of-sample accuracy rate.

Check variable importance

Since we opted for the random forest model, we can extract the variable importance from it. This tells us how the variables helped the model in predicting the class of the data. Higher importance means that the variable is useful to the model.

```r
var_imp <- varImp(Forest, scale = F)$importance
var_imp <- data.frame(variables = row.names(var_imp),
                      importance = var_imp$Overall)
ggplot(aes(x = reorder(variables, importance), y = importance),
       data = var_imp) + geom_bar(stat = 'identity', fill = 'navy') +
        coord_flip() + labs(x = 'Variables',
                            y = 'Importance',
                            title='Random forest variable importance') +
        theme(axis.text = element_text(size = 7),
              axis.title = element_text(size = 15),
              plot.title = element_text(size = 20))
```



Prediction on test set

We may finally apply our model on testing data.

```r
testing <- raw_testing %>%
        select(names(training[,-53]))
dim(testing)
```

```
## [1] 20 52
```

```r
TestRF <- predict(Forest, testing)
TestRF
```

```
##  [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
```