Dumbbell Prediction

Joshua W. Adams

11/4/2020

Introduction

Using tracking devices, it is now possible to collect a large amount of data about personal activity. This project uses data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants in order to predict the manner in which they did the exercise (dumbbell lifts correctly and incorrectly in 5 different ways).

Load Required Packages

We will be using Caret 6.0-86, randomForest 4.6-14, e1071 1.7-4, rattle 5.4.0, and rpart 4.1-15. This document is compiled using R 4.0.3. Also, a random seed will be set for reproducibility. In this example, I am using 1995 as the random seed but you can change the value in the following code if you want to try to get different results.

```
# load packag
library(caret)
library(randomForest)
library(e1071)
library(rattle)
library(rpart)
set.seed(1995)
```

Data Preparation

The data will be pulled from a Cloudfront link, viewable in the below code. It is divided into a training set and a validation set. The training set will be used to create and test a machine learning model. The validation set will be used to test the accuracy of the machine learning model.

```
# Download the training and test data
if(!file.exists("pml-training.csv"))
{
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", "pml-training.cs
}
traindata <- read.csv("pml-training.csv", na.strings = c("NA", ""))
if(!file.exists("pml-testing.csv"))
{
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", "pml-testing.csv
}
validationdata <- read.csv("pml-testing.csv")
```

```
str(traindata)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
##  $ user_name              : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1   : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2   : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484
##  $ cvtd_timestamp         : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" "05/12/20
##  $ new_window             : chr  "no" "no" "no" "no" ...
##  $ num_window             : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt              : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt             : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt               : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt       : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt     : chr  NA NA NA NA ...
##  $ kurtosis_picth_belt    : chr  NA NA NA NA ...
##  $ kurtosis_yaw_belt      : chr  NA NA NA NA ...
##  $ skewness_roll_belt     : chr  NA NA NA NA ...
##  $ skewness_roll_belt.1   : chr  NA NA NA NA ...
##  $ skewness_yaw_belt      : chr  NA NA NA NA ...
##  $ max_roll_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt           : chr  NA NA NA NA ...
##  $ min_roll_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt           : chr  NA NA NA NA ...
##  $ amplitude_roll_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt     : chr  NA NA NA NA ...
##  $ var_total_accel_belt   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x           : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y           : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z           : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x           : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y           : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z           : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x          : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y          : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z          : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm               : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm              : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm                : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm        : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm       : chr  NA NA NA NA ...
##  $ kurtosis_picth_arm      : chr  NA NA NA NA ...
##  $ kurtosis_yaw_arm        : chr  NA NA NA NA ...
##  $ skewness_roll_arm       : chr  NA NA NA NA ...
##  $ skewness_pitch_arm      : chr  NA NA NA NA ...
##  $ skewness_yaw_arm        : chr  NA NA NA NA ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell            : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell  : chr  NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : chr  NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : chr  NA NA NA NA ...
##  $ skewness_roll_dumbbell  : chr  NA NA NA NA ...
##  $ skewness_pitch_dumbbell : chr  NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : chr  NA NA NA NA ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : chr  NA NA NA NA ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : chr  NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

The data has 160 variables, but upon a closer examination, many of these variables contain NA values. These values will need to be eliminated from the tables. The first block is for the training data and the second is for the validation data.

```
# Make a vector of all the columns and the number of NA entries
yesNA = sapply(traindata, function(x) {sum(is.na(x))})
NAColumns = names(yesNA[yesNA > 0])  #Vector with all the columns that has NA values
traindata = traindata[, !names(traindata) %in% NAColumns] # Remove those columms from the training set
```

```
# Remove unnecessary columns in the training data (the first 7 columns)
traindata <- traindata[, !names(traindata) %in% c("X", "user_name", "raw_timestamp_part_1", "raw_timesta

# Repeat the steps with the validation data
validationdata = validationdata[, !names(validationdata) %in% NAColumns]
validationdata <- validationdata[, !names(validationdata) %in% c("X", "user_name", "raw_timestamp_part_
```

Next, the training data will be split into a smaller training set and a testing set. This will be done at a 70/30 ratio and will be called training and testing.

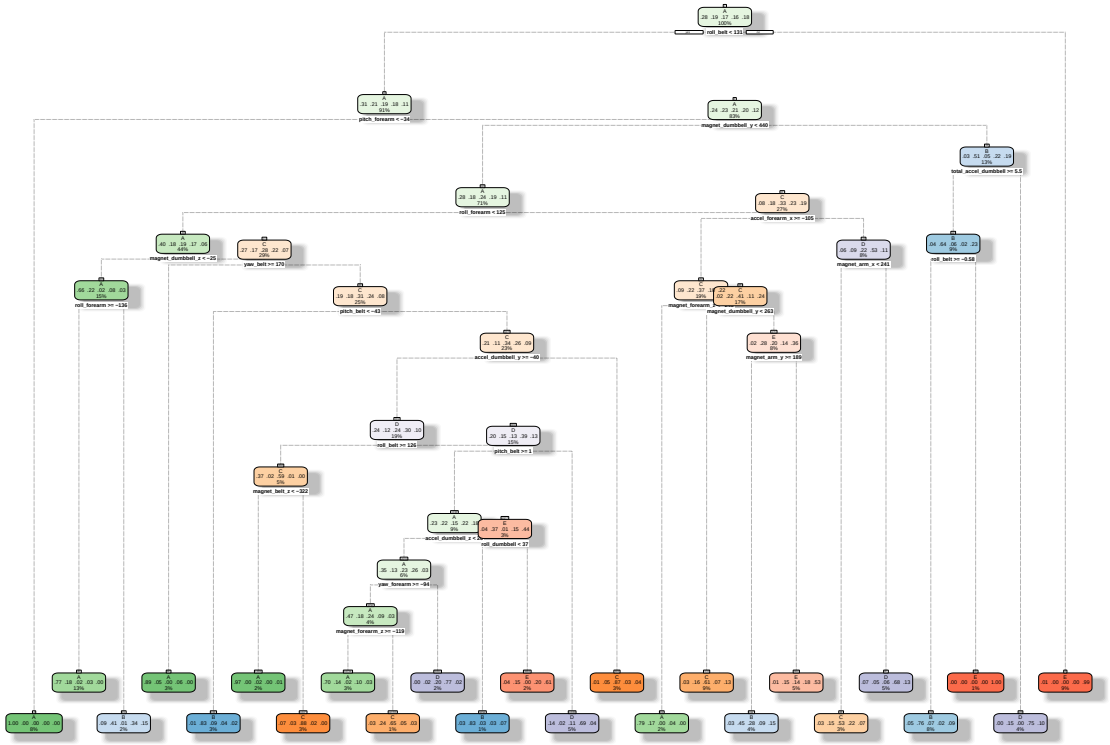
```
inTrain = createDataPartition(y=traindata$classe, p=0.7, list=FALSE)
training = traindata[inTrain,]
testing = traindata[-inTrain,]
```

Model 1

The first model is going to be a classification tree.

```
mod1 <- rpart(classe ~ ., data=training, method="class")
fancyRpartPlot(mod1)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



This is then validated on the testing subset and the accuracy of the model is assessed.

```
predmod1 <- predict(mod1, testing, type = "class")
cm1 <- confusionMatrix(predmod1, as.factor(testing$classe))
cm1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1484  170   25   54   13
##          B   60  691  123   90   96
##          C   39  154  780   77   80
##          D   76   77   64  650   83
##          E   15   47   34   93  810
##
## Overall Statistics
##
##                Accuracy : 0.7502
##                  95% CI : (0.7389, 0.7612)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6837
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8865   0.6067   0.7602   0.6743   0.7486
## Specificity            0.9378   0.9223   0.9280   0.9390   0.9606
## Pos Pred Value         0.8499   0.6519   0.6903   0.6842   0.8108
## Neg Pred Value         0.9541   0.9072   0.9483   0.9364   0.9443
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2522   0.1174   0.1325   0.1105   0.1376
## Detection Prevalence   0.2967   0.1801   0.1920   0.1614   0.1698
## Balanced Accuracy      0.9121   0.7645   0.8441   0.8067   0.8546
```

The classification tree model is providing only a 74.7% accuracy rate, with a potential 95% confidence interval of 73.5 to 75.7%. We should see if we can improve upon this model.

Model 2

A random forest model will be used for the second model. There will be a 3 folds in the cross validation model to help train a potential optimal forest for prediction

```
# instruct train to use 3-fold CV to select optimal tuning parameters
mod2ctrl <- trainControl(method="cv", number=3, verboseIter=F)

# fit model on training subset
mod2 <- train(classe ~ ., data=training, method="rf", trControl=mod2ctrl)

# check parameters
mod2$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: 0.7%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3901    5    0    0    0 0.001280082
## B   16 2636    6    0    0 0.008276900
## C    1   16 2378    1    0 0.007512521
## D    0    0   45 2206    1 0.020426288
## E    0    0    0    5 2520 0.001980198
```

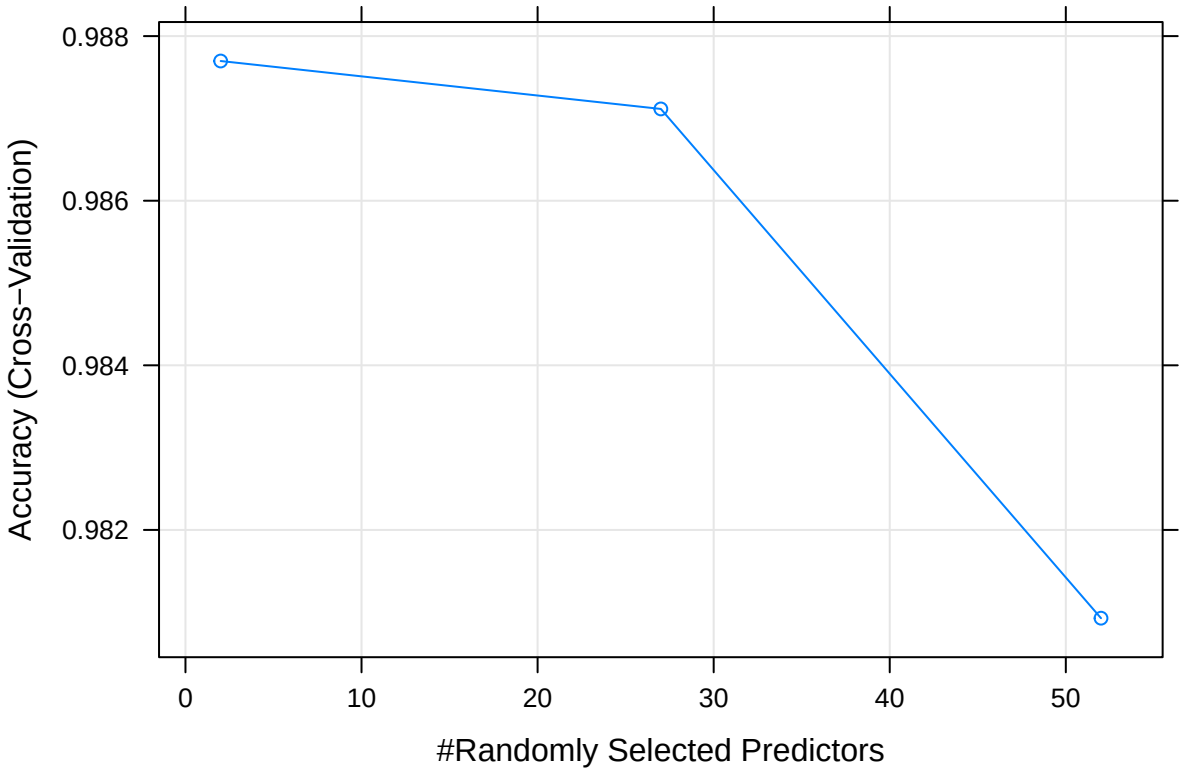
This model has 500 trees and 2 variables at each split. Now this model will be used to predict the data in the testing table, along with using a confusion matrix to demonstrate the accuracy of those predictions.

```
# use model to predict classe in testing subset
predmod2 <- predict(mod2, newdata=testing)

# show confusion matrix to get estimate of out-of-sample error
cm2 <- confusionMatrix(predmod2, as.factor(testing$classe))
mod2accuracy <- cm2$overall["Accuracy"]
mod2accuracy
```

```
##  Accuracy
## 0.9926933
```

```
plot(mod2)
```



The accuracy is 99.4%, thus the predicted accuracy for the out-of-sample error is 0.73%. Model 2 with the random forest should be a great model to use on the validation data.

Predicting validation testing data Finally, we use the model fit on the full training set to predict the label for the observations in the validation testing set.

Model 2 Prediction on Validation Data

```
# predict on validation set
mod2valpred <- predict(mod2, newdata=validationdata)
mod2valpred
```

```
##  [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
```

Conclusion

Model 2 using a random forest predicted the following values on the validation data set.

B A B A A E D B A A B C B A E E A B B B