Week 4 Assignment: Predicting Exercise Classe from Wearable Devices

Author: Rohan Hadvani

Instructions

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).  
The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

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)

Review criteria

1. Your submission for the Peer Review portion should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).
2. Apply your machine learning algorithm to the 20 test cases available in the test data above and submit your predictions in appropriate format to the Course Project Prediction Quiz for automated grading.

Analysis

Environment setup

**library**(caret)

**library**(randomForest)

**if** (!file.exists('train.csv')) {

download.file(url = 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv',

destfile = 'train.csv', method = 'curl', quiet = TRUE)

}

**if** (!file.exists('test.csv')) {

download.file(url = 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv',

destfile = 'test.csv', method = 'curl', quiet = TRUE)

}

trainRaw <- read.csv('train.csv')

testRaw <- read.csv('test.csv')

Preprocessing

1. First look at the data for each column and remove variables unrelated to exercise (column number and time stamps)

str(trainRaw)

## 'data.frame': 19622 obs. of 160 variables:

## $ X : int 1 2 3 4 5 6 7 8 9 10 ...

## $ user\_name : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...

## $ raw\_timestamp\_part\_1 : int 1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 ...

## $ raw\_timestamp\_part\_2 : int 788290 808298 820366 120339 196328 304277 368296 440390 484323 484434 ...

## $ cvtd\_timestamp : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...

## $ new\_window : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...

## $ 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 : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ kurtosis\_picth\_belt : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ kurtosis\_yaw\_belt : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...

## $ skewness\_roll\_belt : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ skewness\_roll\_belt.1 : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ skewness\_yaw\_belt : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...

## $ 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 : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ 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 : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ 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 : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ 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 : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ kurtosis\_picth\_arm : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ kurtosis\_yaw\_arm : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ skewness\_roll\_arm : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ skewness\_pitch\_arm : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ skewness\_yaw\_arm : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ 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 : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ kurtosis\_picth\_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ kurtosis\_yaw\_dumbbell : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...

## $ skewness\_roll\_dumbbell : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ skewness\_pitch\_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ skewness\_yaw\_dumbbell : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...

## $ 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 : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ 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 : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...

## $ amplitude\_roll\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...

## [list output truncated]

train <- trainRaw[, 6:ncol(trainRaw)]

1. Split the data into 70% training and 30% testing set

set.seed(23954)

inTrain <- createDataPartition(y = train$classe, p = 0.7, list = F)

training <- train[inTrain, ]

testing <- train[-inTrain, ]

1. Remove the variables with a lot of similarities

nzv <- nearZeroVar(train, saveMetrics = T)

keepFeat <- row.names(nzv[nzv$nzv == FALSE, ])

training <- training[, keepFeat]

1. Remove the variables with all NAs

training <- training[, colSums(is.na(training)) == 0]

dim(training)

## [1] 13737 54

This is a rather stringent cutoff but there is still >50 features after removal

Model training

1. Set up 5-fold cross validation for training

modCtl <- trainControl(method = 'cv', number = 5)

1. Fit a model with random forests

set.seed(2384)

modRf <- train(classe ~. , data = training, method = 'rf', trControl = modCtl)

* Read the summary of the model built with random forests

modRf$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: 0.2%

## Confusion matrix:

## A B C D E class.error

## A 3906 0 0 0 0 0.000000000

## B 8 2647 3 0 0 0.004138450

## C 0 5 2391 0 0 0.002086811

## D 0 0 6 2245 1 0.003108348

## E 0 0 0 5 2520 0.001980198

* Predict with the validation set and check the confusion matrix and accuracy

predRf <- predict(modRf, newdata = testing)

confusionMatrix(predRf, testing$classe)$table

## Reference

## Prediction A B C D E

## A 1673 10 0 0 0

## B 0 1128 6 0 0

## C 0 1 1020 5 0

## D 0 0 0 959 4

## E 1 0 0 0 1078

confusionMatrix(predRf, testing$classe)$overall[1]

## Accuracy

## 0.9954121

The accuracy is ~99.6% under 5-fold cross validation

1. Fit a model with gradient boosting method

modGbm <- train(classe ~., data = training, method = 'gbm', trControl = modCtl, verbose = F)

* Read the summary of the model built with gbm

modGbm$finalModel

## A gradient boosted model with multinomial loss function.

## 150 iterations were performed.

## There were 53 predictors of which 40 had non-zero influence.

* Predict with the validation set and check the confusion matrix and accuracy

predGbm <- predict(modGbm, newdata = testing)

confusionMatrix(predGbm, testing$classe)$table

## Reference

## Prediction A B C D E

## A 1666 15 0 0 0

## B 8 1114 10 3 4

## C 0 8 1010 11 2

## D 0 2 6 950 9

## E 0 0 0 0 1067

confusionMatrix(predGbm, testing$classe)$overall[1]

## Accuracy

## 0.986746

The accuracy is ~98.8% under 5-fold cross validation

Quiz

Since random forests gives the highest accuracy under the validation set, this model will be selected and used for prediction in the test set

predRfTest <- predict(modRf, newdata = testRaw)

predRfTest

## [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

The gbm model can also be used for prediction and the results can be compared to above

predGbmTest <- predict(modGbm, newdata = testRaw)

table(predRfTest, predGbmTest)

## predGbmTest

## predRfTest A B C D E

## A 7 0 0 0 0

## B 0 8 0 0 0

## C 0 0 1 0 0

## D 0 0 0 1 0

## E 0 0 0 0 3

The two models produce the same results, as shown in the confusion matrix above