Practical Machine Learning Course Project

Samantha Spallone

February 19, 2017

Instructions

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.

Review criteria

What you should submit

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**
- **Why you made the choices you did**

You will also use your prediction model to predict 20 different test cases.

Peer Review Portion

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 :-). Course Project Prediction Quiz Portion

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.

Reproducibility

Due to security concerns with the exchange of R code, your code will not be run during the evaluation by your classmates. Please be sure that if they download the repo, they will be able to view the compiled HTML version of your analysis.

Prediction Assignment Writeup

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 the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset). 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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Input Data

I will load the R packages needed for analysis and download the training and testing data sets.

```
# load the required packages
library(caret); library(rattle); library(rpart); library(rpart.plot)
library(randomForest); library(repmis); library(e1071)

# import the data from the URLs
trainurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
testurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
training <- source_data(trainurl, na.strings = c("NA", "#DIV/0!", ""),
header = TRUE)
testing <- source_data(testurl, na.strings = c("NA", "#DIV/0!", ""),
header = TRUE)

dim(training)
```

```
[1] 19622   160
```

```
dim(testing)
```

```
[1]  20 160
```

The training dataset has 19622 observations and 160 variables, and the testing data set contains 20 observations and the same 160 variables.

Data Cleaning

I will now delete columns (predictors) of the training set that contain any missing values.

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

```
names(training)
```

```
 [1] "V1"                    "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"
```

```
names(testing)
```

```
 [1] "V1"                    "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"      "problem_id"
```

I will remove the first seven columns because these variables are irrelevant for predicting the outcome variable "classe."

```
training_data <- training[, -c(1:7)]
testing_data <- testing[, -c(1:7)]

dim(training_data)

[1] 19622    53

dim(testing_data)

[1] 20 53
```

The cleaned data sets both have 53 columns. The first 52 variables are the same, but the last variable in the training_data is "classe," while the last variable in the testing_data is "problem_id." The training_data still has 19622 rows, and testing_data still has 20 rows.

Data Spliting

I will split the cleaned training set (training_data) into a training set (train) for prediction and a validation set (validate) for computing out of sample error.

```
set.seed(7826)
inTrain <- createDataPartition(training_data$classe, p = 0.7, list =
FALSE)
train <- training_data[inTrain, ]
validate <- training_data[-inTrain, ]
```

Algorithm

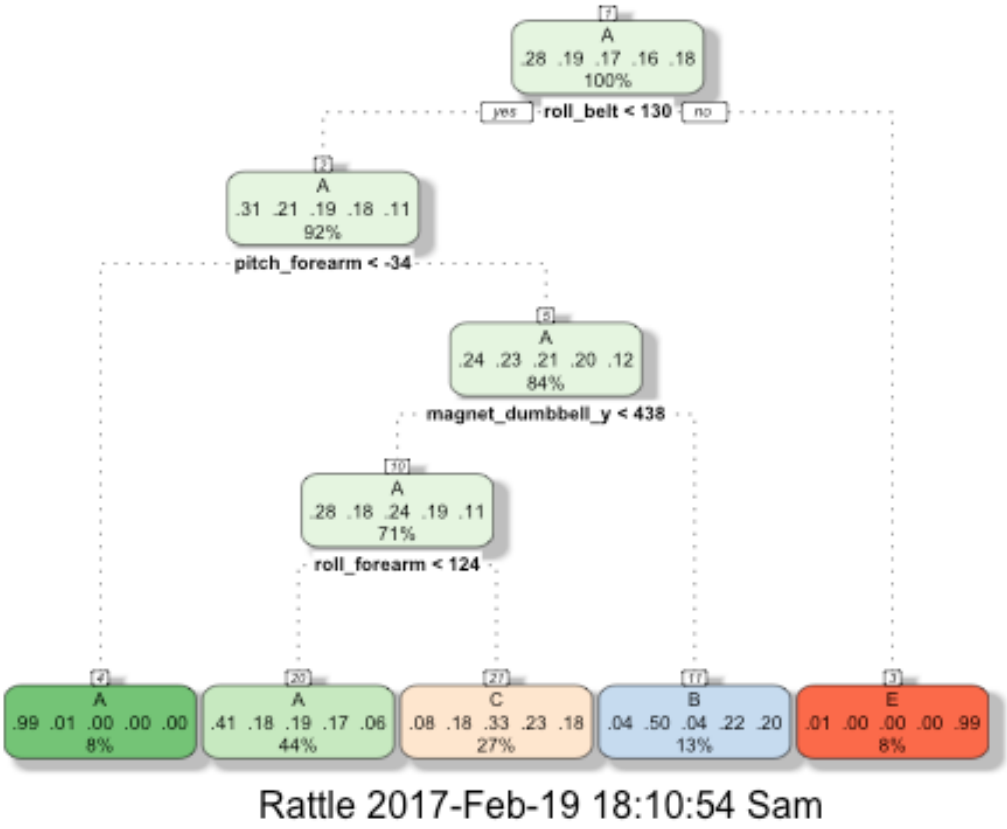
Classification Tree

Below is a 5-fold cross validation. I chose k=5 instead of the default, k=10, to save some computing time. I did not transform any variables.

```r
control <- trainControl(method = "cv", number = 5)
fit_rpart <- train(classe ~ ., data = train, method = "rpart",
trControl = control)
print(fit_rpart, digits = 4)
```

```
CART

13737 samples
   52 predictor
    5 classes: 'A', 'B', 'C', 'D', 'E'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 10989, 10989, 10990, 10989, 10991
Resampling results across tuning parameters:

  cp       Accuracy  Kappa
  0.03723  0.5241    0.38748
  0.05954  0.4144    0.20668
  0.11423  0.3482    0.09762

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was cp = 0.03723.
```

```r
fancyRpartPlot(fit_rpart$finalModel)
```



Now I will predict the outcomes using the validation data set.

```
predict_rpart <- predict(fit_rpart, validate)
confusion_rpart <- confusionMatrix(validate$classe, predict_rpart)
confusion_rpart

Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1544   21  107    0    2
         B  492  391  256    0    0
         C  474   38  514    0    0
         D  436  175  353    0    0
         E  155  138  293    0  496

Overall Statistics

               Accuracy : 0.5004
                 95% CI : (0.4876, 0.5133)
    No Information Rate : 0.5269
    P-Value [Acc > NIR] : 1
```

```
                  Kappa : 0.3464
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.4979  0.51245  0.33749       NA  0.99598
Specificity            0.9533  0.85396  0.88262   0.8362  0.89122
Pos Pred Value         0.9223  0.34328  0.50097       NA  0.45841
Neg Pred Value         0.6303  0.92162  0.79234       NA  0.99958
Prevalence             0.5269  0.12965  0.25879   0.0000  0.08462
Detection Rate         0.2624  0.06644  0.08734   0.0000  0.08428
Detection Prevalence   0.2845  0.19354  0.17434   0.1638  0.18386
Balanced Accuracy      0.7256  0.68321  0.61006       NA  0.94360

accuracy_rpart <- confusion_rpart$overall[1]
accuracy_rpart

 Accuracy
0.5004248
```

The accuracy rate is 0.5 from the confusion matrix, which means that the out of sample error rate is 0.5. The classification tree is not a very good prediction, so I will try the random forest method next.

Random Forest

```
fit_rf <- train(classe ~ ., data = train, method = "rf", trControl =
control)
print(fit_rf, digits = 4)

Random Forest

13737 samples
   52 predictor
    5 classes: 'A', 'B', 'C', 'D', 'E'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 10990, 10990, 10990, 10988, 10990
Resampling results across tuning parameters:

  mtry  Accuracy  Kappa
   2    0.9908    0.9884
  27    0.9906    0.9881
  52    0.9859    0.9821

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
```

Now I will predict the outcomes again using the validation data set.

```
predict_rf <- predict(fit_rf, validate)
confusion_rf <- confusionMatrix(validate$classe, predict_rf)
confusion_rf

Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1673    0    0    0    1
         B    1 1136    2    0    0
         C    0    4 1020    2    0
         D    0    0   21  941    2
         E    0    0    0    2 1080

Overall Statistics

               Accuracy : 0.9941
                 95% CI : (0.9917, 0.9959)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9925
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9994   0.9965   0.9779   0.9958   0.9972
Specificity            0.9998   0.9994   0.9988   0.9953   0.9996
Pos Pred Value         0.9994   0.9974   0.9942   0.9761   0.9982
Neg Pred Value         0.9998   0.9992   0.9953   0.9992   0.9994
Prevalence             0.2845   0.1937   0.1772   0.1606   0.1840
Detection Rate         0.2843   0.1930   0.1733   0.1599   0.1835
Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
Balanced Accuracy      0.9996   0.9979   0.9884   0.9956   0.9984

accuracy_rf <- confusion_rf$overall[1]
accuracy_rf

 Accuracy
0.9940527
```

The accuracy rate is 0.991, which means the out of sample error rate is 0.009. The random forest method is much better at predicting the outcome than the classification tree method.

Evaluation

Prediction on Testing Set

I am now ready to use the random forest method to predict the outcome variable "classe," using the 20 cases in the testing set.

```
predict(fit_rf, testing_data)
```

```
 [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
```