Pratical Machine Learning Project

Yvonne Low

26 April, 2015

Executive Summary

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.

The aim of this report was to use data from accelerometers placed on the belt, forearm, arm, and dumbell of six participants to predict how well they were doing the exercise in terms of the classification in the data.

Open Libraries

```
library(caret, lib.loc="/Users/Yvonne/Documents/R packages/")
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
#library(corrplot, lib.loc="/Users/Yvonne/Documents/R packages/")
#library(kernlab, lib.loc="/Users/Yvonne/Documents/R packages/")
#library(knitr, lib.loc="/Users/Yvonne/Documents/R packages/")
library(rpart, lib.loc="/Users/Yvonne/Documents/R packages/")
library(e1071, lib.loc="/Users/Yvonne/Documents/R packages/")
library(randomForest, lib.loc="/Users/Yvonne/Documents/R packages/")
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

Loading and preprocessing the data

Download input files

```
#training_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
#testing_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
dest_training <- "/Users/Yvonne/Documents/Yvonne Low/Data Science Course/Practical Machine Learning/data
dest_testing <-  "/Users/Yvonne/Documents/Yvonne Low/Data Science Course/Practical Machine Learning/data
#download.file(url=training_url, destfile=dest_training, method="curl")
#download.file(url=testing_url, destfile=dest_testing, method="curl")
```

Treat empty values as NA.

```
df_training <- read.csv(dest_training, na.strings=c("NA",""), header=TRUE)
colnames_train <- colnames(df_training)
df_testing <- read.csv(dest_testing, na.strings=c("NA",""), header=TRUE)
colnames_test <- colnames(df_testing)
```

1 useful step to verify that the column names are identical in the training and test set.

```
all.equal(colnames_train[1:length(colnames_train)-1], colnames_test[1:length(colnames_train)-1])
```

```
## [1] TRUE
```

Count the number of non-NAs in each col.

```
nonNAs <- function(x) {
  as.vector(apply(x, 2, function(x) length(which(!is.na(x)))))
}
# Build vector of missing data or NA columns to drop.
colcnts <- nonNAs(df_training)
drops <- c()
for (cnt in 1:length(colcnts)) {
  if (colcnts[cnt] < nrow(df_training)) {
    drops <- c(drops, colnames_train[cnt])
  }
}
```

There was a lot of NA values in the data which would create a lot of noise for the model. As a result, these columns were removed from the data set. The first eight columns that acted as identifiers for the experiment were also removed. ## Drop NA columns

```
df_training <- df_training[,!(names(df_training) %in% drops)]
df_testing <- df_testing[,!(names(df_testing) %in% drops)]
```

Drop identifier columns in the first 7 columns

```
df_training <- df_training[,8:length(colnames(df_training))]
df_testing <- df_testing[,8:length(colnames(df_testing))]
# Show remaining columns.
colnames(df_training)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
```

```
## [10] "accel_belt_z"          "magnet_belt_x"         "magnet_belt_y"
## [13] "magnet_belt_z"         "roll_arm"              "pitch_arm"
## [16] "yaw_arm"               "total_accel_arm"       "gyros_arm_x"
## [19] "gyros_arm_y"           "gyros_arm_z"           "accel_arm_x"
## [22] "accel_arm_y"           "accel_arm_z"           "magnet_arm_x"
## [25] "magnet_arm_y"          "magnet_arm_z"          "roll_dumbbell"
## [28] "pitch_dumbbell"        "yaw_dumbbell"          "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"      "gyros_dumbbell_y"      "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"      "accel_dumbbell_y"      "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"     "magnet_dumbbell_y"     "magnet_dumbbell_z"
## [40] "roll_forearm"          "pitch_forearm"         "yaw_forearm"
## [43] "total_accel_forearm"   "gyros_forearm_x"       "gyros_forearm_y"
## [46] "gyros_forearm_z"       "accel_forearm_x"       "accel_forearm_y"
## [49] "accel_forearm_z"       "magnet_forearm_x"      "magnet_forearm_y"
## [52] "magnet_forearm_z"      "classe"
```

Check for covariates

```
nsv <- nearZeroVar(df_training, saveMetrics=TRUE)
nsv
```

```
##                      freqRatio percentUnique zeroVar   nzv
## roll_belt             1.101904     6.7781062   FALSE FALSE
## pitch_belt            1.036082     9.3772296   FALSE FALSE
## yaw_belt              1.058480     9.9734991   FALSE FALSE
## total_accel_belt      1.063160     0.1477933   FALSE FALSE
## gyros_belt_x          1.058651     0.7134849   FALSE FALSE
## gyros_belt_y          1.144000     0.3516461   FALSE FALSE
## gyros_belt_z          1.066214     0.8612782   FALSE FALSE
## accel_belt_x          1.055412     0.8357966   FALSE FALSE
## accel_belt_y          1.113725     0.7287738   FALSE FALSE
## accel_belt_z          1.078767     1.5237998   FALSE FALSE
## magnet_belt_x         1.090141     1.6664968   FALSE FALSE
## magnet_belt_y         1.099688     1.5187035   FALSE FALSE
## magnet_belt_z         1.006369     2.3290184   FALSE FALSE
## roll_arm             52.338462    13.5256345   FALSE FALSE
## pitch_arm            87.256410    15.7323412   FALSE FALSE
## yaw_arm              33.029126    14.6570176   FALSE FALSE
## total_accel_arm       1.024526     0.3363572   FALSE FALSE
## gyros_arm_x           1.015504     3.2769341   FALSE FALSE
## gyros_arm_y           1.454369     1.9162165   FALSE FALSE
## gyros_arm_z           1.110687     1.2638875   FALSE FALSE
## accel_arm_x           1.017341     3.9598410   FALSE FALSE
## accel_arm_y           1.140187     2.7367241   FALSE FALSE
## accel_arm_z           1.128000     4.0362858   FALSE FALSE
## magnet_arm_x          1.000000     6.8239731   FALSE FALSE
## magnet_arm_y          1.056818     4.4439914   FALSE FALSE
## magnet_arm_z          1.036364     6.4468454   FALSE FALSE
## roll_dumbbell         1.022388    84.2065029   FALSE FALSE
## pitch_dumbbell        2.277372    81.7449801   FALSE FALSE
## yaw_dumbbell          1.132231    83.4828254   FALSE FALSE
## total_accel_dumbbell  1.072634     0.2191418   FALSE FALSE
```

```
## gyros_dumbbell_x      1.003268     1.2282132   FALSE FALSE
## gyros_dumbbell_y      1.264957     1.4167771   FALSE FALSE
## gyros_dumbbell_z      1.060100     1.0498420   FALSE FALSE
## accel_dumbbell_x      1.018018     2.1659362   FALSE FALSE
## accel_dumbbell_y      1.053061     2.3748853   FALSE FALSE
## accel_dumbbell_z      1.133333     2.0894914   FALSE FALSE
## magnet_dumbbell_x     1.098266     5.7486495   FALSE FALSE
## magnet_dumbbell_y     1.197740     4.3012945   FALSE FALSE
## magnet_dumbbell_z     1.020833     3.4451126   FALSE FALSE
## roll_forearm         11.589286    11.0895933   FALSE FALSE
## pitch_forearm        65.983051    14.8557741   FALSE FALSE
## yaw_forearm          15.322835    10.1467740   FALSE FALSE
## total_accel_forearm   1.128928     0.3567424   FALSE FALSE
## gyros_forearm_x       1.059273     1.5187035   FALSE FALSE
## gyros_forearm_y       1.036554     3.7763735   FALSE FALSE
## gyros_forearm_z       1.122917     1.5645704   FALSE FALSE
## accel_forearm_x       1.126437     4.0464784   FALSE FALSE
## accel_forearm_y       1.059406     5.1116094   FALSE FALSE
## accel_forearm_z       1.006250     2.9558659   FALSE FALSE
## magnet_forearm_x      1.012346     7.7667924   FALSE FALSE
## magnet_forearm_y      1.246914     9.5403119   FALSE FALSE
## magnet_forearm_z      1.000000     8.5771073   FALSE FALSE
## classe                1.469581     0.0254816   FALSE FALSE
```

Create the model

The test data set was split up into training and cross validation sets in a 60:40 ratio in order to train the model and then test it against data it was not specifically fitted to. ## Split the training data again into training and cross validation

```
set.seed(102)
inTrain <- createDataPartition(y = df_training$classe, p = 0.6, list = FALSE)
training <- df_training[inTrain, ]
crossval <- df_training[-inTrain, ]
```

Fit a model to predict the classe using everything else as a predictor

```
model <- randomForest(classe ~ ., data = training)
```

Crossvalidate the model using the remaining 40% of data

The model was then used to classify the remaining 40% of data. The results were placed in a confusion matrix along with the actual classifications in order to determine the accuracy of the model.

```
predictCrossVal <- predict(model, crossval)
confusionMatrix(crossval$classe, predictCrossVal)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2227    5    0    0    0
##          B    3 1511    4    0    0
##          C    0    6 1361    1    0
##          D    0    0   21 1262    3
##          E    0    0    2    5 1435
##
## Overall Statistics
##
##                Accuracy : 0.9936
##                  95% CI : (0.9916, 0.9953)
##     No Information Rate : 0.2842
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9919
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9928   0.9805   0.9953   0.9979
## Specificity            0.9991   0.9989   0.9989   0.9964   0.9989
## Pos Pred Value         0.9978   0.9954   0.9949   0.9813   0.9951
## Neg Pred Value         0.9995   0.9983   0.9958   0.9991   0.9995
## Prevalence             0.2842   0.1940   0.1769   0.1616   0.1833
## Detection Rate         0.2838   0.1926   0.1735   0.1608   0.1829
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9989   0.9958   0.9897   0.9958   0.9984
```

This model yielded a 99.3% prediction accuracy. Again, this model proved very robust and adequete to predict new data.

Prediction

Predict the classes of the original Testing data

A separate data set was then loaded into R and cleaned in the same manner as before. The model was then used to predict the classifications of the 20 results of this new data.

```
predictTest <- predict(model, df_testing)
predictTest
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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

The accuracy of the model is 99.38% giving: B A B A A E D B A A B C B A E E A B B B