Practical Machine Learning Course Project: Predictions with Biometric Datasets

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, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict a representative "classe" variable indicative of performance.

Obtaining, partitioning, and subsetting the dataset to reflect patterns that are likely predictive

```
install.packages("caret", repos="http://cran.rstudio.org")
```

```
## Installing package into 'C:/Users/ghb206/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)
```

```
## package 'caret' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\ghb206\AppData\Local\Temp\RtmpigbPDh\downloaded_packages
```

```
install.packages("e1071", repos="http://cran.rstudio.org")
```

```
## Installing package into 'C:/Users/ghb206/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)
```

```
## package 'e1071' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\ghb206\AppData\Local\Temp\RtmpigbPDh\downloaded_packages
```

```
install.packages("rpart", repos="http://cran.rstudio.org")
```

```
## Installing package into 'C:/Users/ghb206/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)
```

```
##
##   There is a binary version available but the source version is
##   later:
##       binary source needs_compilation
## rpart 4.1-10 4.1-11              TRUE
##
##   Binaries will be installed
## package 'rpart' successfully unpacked and MD5 sums checked
```

```
##
## The downloaded binary packages are in
##  C:\Users\ghb206\AppData\Local\Temp\RtmpigbPDh\downloaded_packages

install.packages("randomForest", repos="http://cran.rstudio.org")

## Installing package into 'C:/Users/ghb206/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)

## package 'randomForest' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\ghb206\AppData\Local\Temp\RtmpigbPDh\downloaded_packages

install.packages("knitr", repos="http://cran.rstudio.org")

## Installing package into 'C:/Users/ghb206/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)

## package 'knitr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\ghb206\AppData\Local\Temp\RtmpigbPDh\downloaded_packages

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(e1071)
library(rpart)
library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

library(knitr)



#Get the train and test data

mlTest <-"C:/Users/ghb206/Documents/DataSciTrack_JHU/mlTest.csv"
mlTestURL <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
```

```
testing.csv"

mlTrain <-"C:/Users/ghb206/Documents/DataSciTrack_JHU/mlTrain.csv"
mlTrainURL <-
"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"


Test<-download.file(mlTestURL, mlTest, mode = "wb")
mlTestData<-read.csv("C:/Users/ghb206/Documents/DataSciTrack_JHU/mlTest.csv",
header = TRUE)

Train<-download.file(mlTrainURL, mlTrain, mode = "wb")
mlTrainData<-
read.csv("C:/Users/ghb206/Documents/DataSciTrack_JHU/mlTrain.csv", header =
TRUE)

#Partion the data per the popular 70/30 split in favor of training data

holder <- createDataPartition(mlTrainData$classe, p=0.7, list=FALSE)
sevSplTrainData <- mlTrainData[holder, ]
thirSplTestData <- mlTrainData[-holder, ]

#confirm with dim() that all matches
dim(mlTrainData)

## [1] 19622   160

dim(sevSplTrainData)

## [1] 13737   160

dim(thirSplTestData)

## [1] 5885  160

#Now examine the completeness of the data

naTrain <-sapply(sevSplTrainData, function(x) sum(length(which(is.na(x)))))

#naTrain <- data.frame(naTrain)
#Note, this is supressed for brevity, but there are many NA's

naTest <-sapply(thirSplTestData, function(x) sum(length(which(is.na(x)))))

#naTest <- data.frame(naTest)
#As above, so the problematic columns will be removed

#Two datasets will be made...one dataset will include all of the
#columns that lack NA values and factorized categories and the
#other will include only the
```

```
#summaries (totals per category)...both will be tested with various
#ML algorithms but only summaries of the better will be provided
#the goal is to see how a vastly stripped dataset performs relative
#to a large, near complete presentation

#Again, the first will include all of the non NA containing columns
#and further filter by items being read as factors

scaleSevSplTrainData<-sevSplTrainData[ , colSums(is.na(sevSplTrainData)) ==
0]
scaleThirSplTestData<-thirSplTestData[ , colSums(is.na(thirSplTestData)) ==
0]


numScaleSevSplTrainData <- sapply(scaleSevSplTrainData, is.numeric)
numScaleThirSplTestData <- sapply(scaleThirSplTestData, is.numeric)

newNumScaleSevSplTrainData<-scaleSevSplTrainData[,numScaleThirSplTestData]
newNumScaleThirSplTestData<-scaleThirSplTestData[,numScaleThirSplTestData]

#Re-shaping the more complete dataset

finalScaleSplTrainData<-
cbind(sevSplTrainData$user_name,sevSplTrainData$classe,
newNumScaleSevSplTrainData)

finalScaleSplTestData<-
cbind(thirSplTestData$user_name,thirSplTestData$classe,
newNumScaleThirSplTestData)

colnames(finalScaleSplTrainData)[1] <- "user_name"
colnames(finalScaleSplTrainData)[2] <- "classe"

colnames(finalScaleSplTestData)[1] <- "user_name"
colnames(finalScaleSplTestData)[2] <- "classe"

finalScaleSplTrainData$X <-NULL
finalScaleSplTestData$X <-NULL

#The second will be just the "total" categories


scaleTwoSevSplTrainData<-sevSplTrainData[ , grep("total",
colnames(sevSplTrainData))]

scaleTwoThirSplTestData<-thirSplTestData[ , grep("total",
colnames(thirSplTestData))]
```

```
#More supressed output...the var column is all NA per the below so dropped
#sum(is.na(scaleTwoSevSplTrainData$var_total_accel_belt))
#sum(is.na(scaleTwoThirSplTestData$var_total_accel_belt))
#NOTE:  "scaledTwo" reflects the same column sum as "scaled"

scaleTwoSevSplTrainData$var_total_accel_belt<-NULL
scaleTwoThirSplTestData$var_total_accel_belt<-NULL

#Re-shape smaller "total" dataset after the grep command

scaleTwoSevSplTrainData<-
cbind(sevSplTrainData$user_name,sevSplTrainData$classe,
scaleTwoSevSplTrainData)


scaleTwoThirSplTestData<-
cbind(thirSplTestData$user_name,thirSplTestData$classe,
scaleTwoThirSplTestData)

colnames(scaleTwoSevSplTrainData)[1] <- "user_name"
colnames(scaleTwoSevSplTrainData)[2] <- "classe"

colnames(scaleTwoThirSplTestData)[1] <- "user_name"
colnames(scaleTwoThirSplTestData)[2] <- "classe"


#The two data partitions are formatted and ready for ML algorithm
```

Performing the ML analysis (Random Forest)

```
#Test the minimal set first with "random forest"


set.seed(3730977)
randForestModel1 <- randomForest(classe ~ ., data=scaleTwoSevSplTrainData)
randForestPrediction1 <- predict(randForestModel1, scaleTwoThirSplTestData,
type = "class")
randForestConfMatrix1 <- confusionMatrix(randForestPrediction1,
scaleTwoThirSplTestData$classe)

#randForestConfMatrix1 output not shown as the model was only
#minimally accurate


#Now test the assumed "more robust" set with "random forest"


randForestModel2 <- randomForest(classe ~ ., data=finalScaleSplTrainData)
randForestPrediction2 <- predict(randForestModel2, finalScaleSplTestData,
```

```
type = "class")
randForestConfMatrix <- confusionMatrix(randForestPrediction2,
finalScaleSplTestData$classe)

randForestConfMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    2    0    0
##          C    0    0 1021    4    0
##          D    0    0    3  960    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9983
##                  95% CI : (0.9969, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9951   0.9959   1.0000
## Specificity            0.9998   0.9996   0.9992   0.9994   1.0000
## Pos Pred Value         0.9994   0.9982   0.9961   0.9969   1.0000
## Neg Pred Value         1.0000   0.9998   0.9990   0.9992   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1735   0.1631   0.1839
## Detection Prevalence   0.2846   0.1937   0.1742   0.1636   0.1839
## Balanced Accuracy      0.9999   0.9994   0.9972   0.9976   1.0000
```

- The second random forest based model (built with far more data) performed much better in terms of accuracy. For added assurance, the Kappa, which is a measure of observed relative to expected accuracy is 1.0.

Performing the second ML analysis (Decison Tree)

```
set.seed(37485)
dTreeModel1 <- rpart(classe ~ ., data=scaleTwoSevSplTrainData,
method="class")
dTreePfrediction1 <- predict(dTreeModel1, scaleTwoThirSplTestData, type =
"class")
dTreeConfMatrix1 <- confusionMatrix(dTreePfrediction1,
scaleTwoThirSplTestData$classe)
```

```
#dTreeConfMatrix1 / supressed due to low accuracy


dTreeModel2 <- rpart(classe ~ ., data=finalScaleSplTrainData, method="class")
dTreePfrediction2 <- predict(dTreeModel2, finalScaleSplTestData, type =
"class")
dTreeConfMatrix2 <- confusionMatrix(dTreePfrediction2,
finalScaleSplTestData$classe)
dTreeConfMatrix2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1585  141    4   19   17
##          B   42  794   36   35   72
##          C    5   53  887   65   75
##          D   36   93   32  719   61
##          E    6   58   67  126  857
##
## Overall Statistics
##
##                Accuracy : 0.8228
##                  95% CI : (0.8128, 0.8324)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7755
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9468   0.6971   0.8645   0.7459   0.7921
## Specificity            0.9570   0.9610   0.9593   0.9549   0.9465
## Pos Pred Value         0.8975   0.8110   0.8175   0.7641   0.7693
## Neg Pred Value         0.9784   0.9297   0.9710   0.9504   0.9528
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2693   0.1349   0.1507   0.1222   0.1456
## Detection Prevalence   0.3001   0.1664   0.1844   0.1599   0.1893
## Balanced Accuracy      0.9519   0.8291   0.9119   0.8504   0.8693
```

- Once again, the expanded model had a resasonably high level of accuracy, and a strong Kappa statistic. Since the random forest model performed better, however, it was chosen for the predictive exercise. Of course, it should be pointed out that these results are seed specific and driven by the underlying dataset. In addition, the 70/30 partition could play a role as well.

Now get the predictions on the small test set (not derived from the original training partition)

```
#in order to predict, the file needs to resemble the format in out best model

scale1MLTestData<-mlTestData[ , colSums(is.na(mlTestData)) == 0]

scale2MLTestData <- sapply(scale1MLTestData, is.numeric)

scale3MLTestData<-scale1MLTestData[,scale2MLTestData]

#newNumScaleSevSplTrainData<-scaleSevSplTrainData[,numScaleThirSplTestData]

scale4MLTestData<-cbind(mlTestData$user_name,
scale3MLTestData)

colnames(scale4MLTestData)[1] <- "user_name"

scale4MLTestData$X <- NULL


randForestModelPredict <- predict(randForestModel2, scale4MLTestData, type =
"class")
randForestModelPredict

##  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

#These results were able to correctly predict the "classe" category for
#each tested instance
```